# MATLAB®

The Language of Technical Computing

Computation

Visualization

Programming

Release Notes for Release 12.1

The MathWorks

**How to Contact The MathWorks:**

| | |
|---|---|
| `www.mathworks.com` | Web |
| `comp.soft-sys.matlab` | Newsgroup |

| | |
|---|---|
| `support@mathworks.com` | Technical support |
| `suggest@mathworks.com` | Product enhancement suggestions |
| `bugs@mathworks.com` | Bug reports |
| `doc@mathworks.com` | Documentation error reports |
| `service@mathworks.com` | Order status, license renewals, passcodes |
| `info@mathworks.com` | Sales, pricing, and general information |

508-647-7000 Phone

508-647-7001 Fax

The MathWorks, Inc. Mail
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

*Release Notes for Release 12.1*

© COPYRIGHT 2001 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by or for the federal government of the United States. By accepting delivery of the Program, the government hereby agrees that this software qualifies as "commercial" computer software within the meaning of FAR Part 12.212, DFARS Part 227.7202-1, DFARS Part 227.7202-3, DFARS Part 252.227-7013, and DFARS Part 252.227-7014. The terms and conditions of The MathWorks, Inc. Software License Agreement shall pertain to the government's use and disclosure of the Program and Documentation, and shall supersede any conflicting contractual terms or conditions. If this license fails to meet the government's minimum needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to MathWorks.

MATLAB, Simulink, Stateflow, Handle Graphics, and Real-Time Workshop are registered trademarks, and Target Language Compiler is a trademark of The MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Printing History: June 2001 First printing New for Release 12.1

# Contents

## 1 Introducing Release 12.1


**What's New in Release 12.1?** . . . . . **1-2**
Updated Products . . . . . **1-2**
New Products . . . . . **1-16**

**Platform Limitations** . . . . . **1-17**
Differences for the HP and IBM Platforms . . . . . **1-17**
Problems Entering Accented Characters on Linux . . . . . **1-18**
Problems Entering Accented Characters on UNIX Platforms **1-18**

**Installation Notes** . . . . . **1-19**
Maintaining Preference Files After Uninstalling . . . . . **1-19**
Correcting the Documentation Location . . . . . **1-19**

**Linux Distribution Notes** . . . . . **1-20**

**For More Information About What's New in Release 12.1** **1-21**
Printable Version of the Release Notes . . . . . **1-21**

**MATLAB Software Acknowledgments** . . . . . **1-23**


## 2 MATLAB 6.1 Release Notes


**New Features** . . . . . **2-2**
Development Environment Features . . . . . **2-2**

Mathematics Features . . . . . 2-5
Programming and Data Types Features . . . . . 2-8
Graphics Features . . . . . 2-10
OpenGL Renderer Feature — Microsoft Windows . . . . . 2-11
External Interfaces/API Features . . . . . 2-12
Creating Graphical User Interfaces — GUIDE . . . . . 2-17

**Major Bug Fixes** . . . . . 2-18
Development Environment . . . . . 2-18
Mathematics . . . . . 2-18

**Platform Limitations** . . . . . 2-20
Patch Required for HP-UX 11.0 . . . . . 2-20
Mathematics Limitations . . . . . 2-20
Graphics and Printing Limitations . . . . . 2-21
GUIDE Limitations . . . . . 2-21
You May Need to Overwrite the MATLAB Default Choice of BLAS . . . . . 2-22

**Upgrading from an Earlier Release** . . . . . 2-26
Development Environment Issues . . . . . 2-26
Mathematics Issues . . . . . 2-27
Programming and Data Types Issues . . . . . 2-28
Graphics Issue . . . . . 2-29
External Interfaces/API Issues . . . . . 2-30

**Known Software and Documentation Problems** . . . . . 2-32
Development Environment Problems . . . . . 2-32
Documentation Updates . . . . . 2-33

## 3 Simulink 4.1 Release Notes

**New Features** . . . . . 3-2
Simulink Editor . . . . . 3-2
Modeling Enhancements . . . . . 3-4
Simulink Debugger . . . . . 3-7

Block Library . . . . . 3-8

**Bug Fixes** . . . . . 3-10

**Platform Limitations for HP and IBM** . . . . . 3-12

**Upgrading from an Earlier Release** . . . . . 3-13
Running Simulink 4.1 Models in Simulink 4.0 . . . . . 3-13
Simulink Block Library Reorganization . . . . . 3-13
Direct Feedthrough Compensation Deprecated . . . . . 3-13
S-Functions Sorted Like Built-In Blocks . . . . . 3-14
Added Latched Triggered Subsystems . . . . . 3-14
Self-Triggering Subsystems Are No Longer Allowed . . . . . 3-14
Improved Invalid Model Configuration Diagnostics . . . . . 3-15

**Known Software Problems** . . . . . 3-16
Accelerator Mode Does Not Support Inline Parameters . . . . . 3-16
Turn the New Wrap Lines Option Off . . . . . 3-16

# 4 Stateflow 4.1 Release Notes

**New Features** . . . . . 4-2
Smart Transitions . . . . . 4-2
Search & Replace Tool Enhancements . . . . . 4-3
Stateflow Chart Notes . . . . . 4-3
Model Coverage Tool . . . . . 4-3

**Major Bug Fixes** . . . . . 4-4
Editing Crossing Transitions out of Grouped Subcharts . . . . . 4-4
Disabled and Restored Library Chart Links . . . . . 4-4
Too Many Action Statements During Simulation . . . . . 4-4
False State Inconsistency Runtime Error . . . . . 4-4
MATLAB Variables Improperly Overwritten . . . . . 4-4
Target Options Fields Overwritten . . . . . 4-5
Transitions Assertions . . . . . 4-5
Build Failures with Custom Code . . . . . 4-5

**Upgrading from an Earlier Release** . . . . . **4-6**
Transition Actions into Junctions Disallowed . . . . . **4-6**

# 5 Real-Time Workshop 4.1 Release Notes

**Release Summary** . . . . . **5-2**

**New Features** . . . . . **5-3**
Block Reduction Option On by Default . . . . . **5-3**
Buffer Reuse Code Generation Option . . . . . **5-3**
Build Directory Validation . . . . . **5-4**
Build Subsystem Enhancements . . . . . **5-4**
C API for Parameter Tuning Documented . . . . . **5-4**
Code Readability Improvements . . . . . **5-5**
Control Flow Blocks Support . . . . . **5-5**
Expression Folding . . . . . **5-5**
External Mode Enhancements . . . . . **5-6**
Generate Comments Option . . . . . **5-6**
Include System Hierarchy in Identifiers . . . . . **5-7**
Rapid Simulation Target Supports Inline Parameters . . . . . **5-7**
S-Function Target Enhancements . . . . . **5-7**
Storage Classes for Block States . . . . . **5-7**
Support for tilde (~) in Filenames on UNIX Platforms . . . . . **5-8**
Target Language Compiler 4.1 . . . . . **5-8**

**Bug Fixes** . . . . . **5-11**
Block Reduction Crash Fixed . . . . . **5-11**
Build Subsystem Gives Better Error Message for Function Call Subsystems . . . . . **5-11**
Check Consistency of Parameter Storage Class and Type Qualifier . . . . . **5-11**
Code Optimization for Unsigned Saturation and DeadZone Blocks . . . . . **5-11**
Correct Code Generation of Fixed-Point Blockset Blocks in DSP Blockset Models . . . . . **5-12**

Correct Compilation with Green Hills and DDI Compilers . . . . . 5-12
Fixed Build Error with Models Having Names Identical to Windows NT Commands . . . . . 5-12
Fixed Error Copying Custom Code Blocks . . . . . 5-12
Fixed Error in commonmaplib.tlc . . . . . 5-13
Fixed Name Clashes with Run-Time Library Functions . . . . . 5-13
Improved Handling of Sample Times . . . . . 5-13
Look-Up Table (n-D) Code Generation Bug Fix . . . . . 5-13
Parenthesize Negative Numerics in Fcn Block Expressions . . 5-13
Removed Unnecessary Warnings and Declarations from Generated Code . . . . . 5-14
Retain .rtw File Option Now Works in Accelerator Mode . . . . 5-14
S-Function Target Memory Allocation Bug Fix . . . . . 5-14

**Platform Limitations for HP and IBM** . . . . . 5-15
Tunable Parameters Dialog Box . . . . . 5-15

**Upgrading from an Earlier Release** . . . . . 5-16
RTWInfo Property Changes . . . . . 5-16
S-Function Target MEX-Files Must Be Rebuilt . . . . . 5-17
TLC Compatibility Issues . . . . . 5-17

**Known Software Problem** . . . . . 5-19
Turn the New Wrap Lines Option Off . . . . . 5-19

## 6 CDMA Reference Blockset 1.1 Release Notes

**New Features** . . . . . 6-2
Frame-Based Signal Support . . . . . 6-2

**Upgrading from an Earlier Release** . . . . . 6-3

## 7 Communications Blockset 2.0.1 Release Notes


**New Features** . . . . . 7-2
Setting Simulink Preferences Automatically . . . . . 7-2
Converting Between Bipolar and Unipolar Signals . . . . . 7-2
Choosing Seeds for Random-Output Blocks . . . . . 7-2
Using Error Counts to Control Simulation Duration . . . . . 7-3
Choosing the Algorithm for Integrator Blocks . . . . . 7-3

**Major Bug Fixes** . . . . . 7-4
Bug Fixes Incorporated from Release 12.0 . . . . . 7-4

**Upgrading from an Earlier Release** . . . . . 7-5

**Known Software and Documentation Problems** . . . . . 7-6
Code Generation Limitations . . . . . 7-6
Limited Frame and Matrix Support . . . . . 7-6
Errors in Printed Version of Documentation . . . . . 7-7


## 8 Control System Toolbox 5.1 Release Notes


**New Features** . . . . . 8-2
SISO Design Tool Enhancements . . . . . 8-2
Algorithmic Enhancements . . . . . 8-3

**Platform Limitations for HP and IBM** . . . . . 8-4
Preference and Property Editors . . . . . 8-4
SISO Design Tool . . . . . 8-4

## 9 Data Acquisition Toolbox 2.1 Release Notes


**New Features** . . . . . 9-2
Additional National Instruments Hardware Support . . . . . 9-2
High Resolution Sound Card Support . . . . . 9-2

**Upgrading from an Earlier Release** . . . . . 9-3
Events, Callbacks, and Function Handles . . . . . 9-3
Vendor Drivers . . . . . 9-3
Removed and Obsolete Functions . . . . . 9-4
Two daqhwinfo Fields Removed . . . . . 9-4

**Known Software and Documentation Problems** . . . . . 9-5
High Resolution Sound Cards . . . . . 9-5
The putdata Function . . . . . 9-5
The daqread Function . . . . . 9-5
Vendor Limitations . . . . . 9-5


## 10 Database Toolbox 2.2 Release Notes


**New Features** . . . . . 10-2
Retrieve Data as numeric or structure Format . . . . . 10-2
Export Numeric or Structure Data . . . . . 10-2
Specify Error Handling Preference . . . . . 10-2
LONGCHAR Data Type Supported . . . . . 10-3

**Platform Limitations for HP and Linux** . . . . . 10-4
logintimeout Not Supported on Linux . . . . . 10-4
No HP 10.2 Support . . . . . 10-4

## 11 Datafeed Toolbox 1.3 Release Notes

**New Features** . . . . . **11-2**
Hyperfeed Support Added . . . . . **11-2**
Hyperfeed Function Summary . . . . . **11-2**

**FT Interactive Data Server** . . . . . **11-3**

## 12 Developer's Kit for Texas Instruments™ DSP 1.1 Release Notes

**Introduction to Developer's Kit for Texas Instruments™ DSP** . . . . . **12-2**

**New Features** . . . . . **12-4**
New Link Functions for CCS IDE . . . . . **12-4**
New Interface Between the Developer's Kit and FDATool . . . **12-5**
Unified CCS IDE and C6701 EVM Targeting . . . . . **12-5**
New Run-Time Libraries . . . . . **12-6**
Architecture Change in CCSDSP Object . . . . . **12-6**
Updated Function writemsg Help Text . . . . . **12-7**

**Major Bug Fixes** . . . . . **12-8**

**Upgrading from an Earlier Release** . . . . . **12-9**
New Libraries at Run-Time . . . . . **12-9**
CCSDSP Changes . . . . . **12-9**
Unified Targeting . . . . . **12-9**

**Known Software and Documentation Problems** . . . . . **12-10**
Operating Systems . . . . . **12-10**
Linking . . . . . **12-10**
Targeting . . . . . **12-12**
Demos . . . . . **12-14**
Datatypes . . . . . **12-14**

General Target Operations . . . . . 12-15
Documentation . . . . . 12-15

## 13 DSP Blockset 4.1 Release Notes

**New Features** . . . . . 13-2
New Digital Filter Design Block . . . . . 13-2
Major Individual Block Enhancements . . . . . 13-3
Blocks with Enhanced Data Type Support . . . . . 13-5
Blocks with Tunability Enhancements . . . . . 13-7
Enhancements to Blocks Implemented with S-Functions . . . . 13-7
Real-Time Workshop Generated C Code Enhancements . . . . 13-8
Changes to Filtering Block Libraries . . . . . 13-9
New and Renamed DSP Blockset Directories . . . . . 13-10

**Major Bug Fixes** . . . . . 13-11
Real-Time Workshop C Code Generated from DSP Blockset 13-11
Nontunable Block Parameters . . . . . 13-11
Chirp Block . . . . . 13-11
Matrix Viewer Block . . . . . 13-11
Random Source Block . . . . . 13-12
Stack and Queue Blocks . . . . . 13-12
Window Function Block . . . . . 13-12

**Platform Limitations for HP and IBM** . . . . . 13-13

**Upgrading from an Earlier Release** . . . . . 13-14

**Known Software Problems** . . . . . 13-15
Real-Time Workshop C Code Generation Limitations . . . . . 13-15
Nonzero Simulation Start Times Not Supported . . . . . 13-15
Triggered and Enabled Subsystem Support . . . . . 13-16
Digital Filter Design Block Limitations . . . . . 13-16
Magnitude FFT Block Limitations . . . . . 13-17

## 14 Excel Link 1.1.3 Release Notes

**New Features** . . . . . **14-2**
License Manager Control . . . . . **14-2**
Additional Error Detection . . . . . **14-2**

**Known Software Problems** . . . . . **14-3**
Enabling the MATLAB Desktop . . . . . **14-3**

## 15 Filter Design Toolbox 2.1 Release Notes

**New Features** . . . . . **15-2**
New Adaptive Filtering Functions . . . . . **15-2**
New FIR Filter Design Functions . . . . . **15-3**
New Filter Transformation Functions . . . . . **15-3**
Transformations Option in FDATool . . . . . **15-4**
New Analysis Method . . . . . **15-4**
New Context-Sensitive Help for Quantization . . . . . **15-5**

**Platform Limitations for HP 10.2** . . . . . **15-6**

**Upgrading from an Earlier Release** . . . . . **15-7**
Obsolete Functions in Version 2.0 . . . . . **15-7**

**Known Software and Documentation Problems** . . . . . **15-8**
Switching Between Design and Quantization Modes in FDATool . . . . . **15-8**
Help for Filter Transformations in FDATool . . . . . **15-8**

## 16 Financial Time Series Toolbox 1.1 Release Notes


**New Features** . . . . . 16-2
New Functions . . . . . 16-2
Financial Time Series GUI . . . . . 16-2

**Major Changes to Functions** . . . . . 16-3


## 17 Fixed-Point Blockset 3.1 Release Notes


**New Features** . . . . . 17-2
New Libraries . . . . . 17-2
New Blocks . . . . . 17-3
Renamed Blocks . . . . . 17-8
New Functions . . . . . 17-9
Data Type Support . . . . . 17-10
Changes to Functionality . . . . . 17-10

**Known Documentation Problem** . . . . . 17-11
Error in Printed Version of Documentation . . . . . 17-11


## 18 Image Processing Toolbox 3.1 Release Notes


**New Features** . . . . . 18-2
Morphology . . . . . 18-3
Spatial Transformations . . . . . 18-4
Image Registration . . . . . 18-5
Integer Image Arithmetic . . . . . 18-6
Integer Image Filtering . . . . . 18-7
Deconvolution/Deblurring . . . . . 18-7
Support for DICOM Files . . . . . 18-8
Miscellaneous . . . . . 18-8

New Demos . . . . . 18-9

**Major Bug Fixes** . . . . . 18-11

## 19 Instrument Control Toolbox 1.1 Release Notes

**New Features** . . . . . 19-2
VXI Block and FIFO Read Operations . . . . . 19-2
Freeing the Serial Port on Windows Platforms . . . . . 19-2
Data Parsing . . . . . 19-2
binblock Data Transfer . . . . . 19-3
Tektronix VISA Support . . . . . 19-3
Enhancements to Existing Functions and Properties . . . . . 19-3

**Vendor Driver Requirements and Limitations** . . . . . 19-4
Tektronix VISA Limitations . . . . . 19-4

**Upgrading from an Earlier Release** . . . . . 19-5
Events, Callbacks, and Function Handles . . . . . 19-5

## 20 MATLAB C/C++ Math Library 2.2 Release Notes

**New Features** . . . . . 20-2
New Math Library Functions . . . . . 20-2

**Upgrading from an Earlier Release** . . . . . 20-4
Recompile Your Programs . . . . . 20-4
Changed Features . . . . . 20-4

**Known Software and Documentation Problems** . . . . . 20-6
Documentation Error . . . . . 20-6

## 21 MATLAB Runtime Server 6.1 Release Notes

**New Features** . . . . . 21-2
Configuration File Generation Tool . . . . . 21-2

**Upgrading from an Earlier Release** . . . . . 21-3
Repeat Compiling, Packaging, and Testing Process . . . . . 21-3

**Known Documentation Problems** . . . . . 21-4

## 22 Neural Network Toolbox 4.0.1 Release Notes

**Major Bug Fixes** . . . . . 22-2
Proper Use of Validation and Test Data . . . . . 22-2
Correct Version 3 Object Loading . . . . . 22-2

## 23 Optimization Toolbox 2.1.1 Release Notes

**New Features** . . . . . 23-2
Support for Large Problems that Are Not Well-Scaled . . . . . 23-2

**Major Bug Fixes** . . . . . 23-3

**Upgrading from an Earlier Release** . . . . . 23-4

24 Power System Blockset 2.2 Release Notes

**New Features** . . . . . **24-2**
Powergui Enhancements . . . . . **24-2**
New powerlib_extras Documentation . . . . . **24-2**

25 Real-Time Workshop Ada Coder 4.1 Release Notes

**Release Summary** . . . . . **25-2**

26 Real-Time Workshop Embedded Coder 2.0

**Release Summary** . . . . . **26-2**

**New Features** . . . . . **26-3**
Custom Storage Classes for Data Objects . . . . . **26-3**
Enhanced Code Generation Options . . . . . **26-4**
Virtualized Output Ports Optimization . . . . . **26-4**
Improved HTML Code Generation Report . . . . . **26-4**

27 Requirements Management Interface 1.0.2 Release Notes

**New Features** . . . . . **27-2**
Support for the Latest DOORS Release . . . . . **27-2**

## 28 Signal Processing Toolbox 5.1 Release Notes


**New Features** . . . . . **28-2**
Filter Visualization Tool . . . . . **28-2**
New Windows . . . . . **28-3**
bitrevorder . . . . . **28-4**
downsample/upsample . . . . . **28-4**
filternorm . . . . . **28-4**
goertzel . . . . . **28-4**
New Signal Demo . . . . . **28-4**

**Enhancements** . . . . . **28-5**
FDATool Enhancements . . . . . **28-5**
SPTool Enhancement . . . . . **28-5**
Enhanced Functions . . . . . **28-6**

**Major Bug Fixes** . . . . . **28-7**
Bug Fixes Incorporated from Release 12.0 . . . . . **28-7**

**Platform Limitations** . . . . . **28-8**
FDATool — 640 x 480 Resolution . . . . . **28-8**

**Upgrading from an Earlier Release** . . . . . **28-9**
Changes to remez . . . . . **28-9**

**Known Software Problem** . . . . . **28-10**
FDATool and FVTool Pole/Zero Plot . . . . . **28-10**
FDATool and FVTool Magnitude and Phase Plot . . . . . **28-10**


## 29 Simulink Performance Tools 1.1 Release Notes


**New Features** . . . . . **29-2**
Model Coverage Tool . . . . . **29-2**
Graphical Merge Tool . . . . . **29-4**

**Platform Limitations for HP and IBM** . . . . . **29-6**
Limited Support Model Coverage Tool . . . . . **29-6**
Graphical Merge Tool Not Supported . . . . . **29-6**

## 30 Simulink Report Generator 1.2 Release Notes

**New Features** . . . . . **30-2**
Block Type: Look-Up Table Component Reports on N-D Look-Up Tables . . . . . **30-2**
Real-Time Workshop Requirement Removed for Some Components . . . . . **30-2**
Image Titles for Snapshot Components . . . . . **30-3**

**Upgrading from an Earlier Release** . . . . . **30-4**
Use of Existing SGML Files . . . . . **30-4**

## 31 Stateflow Coder 4.1 Release Notes

**New Features** . . . . . **31-2**
Single-Precision Constants in Code Generation . . . . . **31-2**

**Major Bugs Fixes** . . . . . **31-3**
Code Generation for Default Transitions in Parallel States . . **31-3**
Code Generation for Double-Precision Whole Numbers . . . . . **31-3**

## 32 System Identification Toolbox 5.0.1 Release Notes

**Upgrading from an Earlier Release** . . . . . **32-2**
Syntax of data Function . . . . . **32-2**

Theta Models No Longer Supported . . . . . 32-2

## 33 Virtual Reality Toolbox 2.0 Release Notes

**Introduction to the Virtual Reality Toolbox** . . . . . 33-2
Simulink Interface . . . . . 33-2
MATLAB Interface . . . . . 33-2
Virtual Reality Modeling Language (VRML) . . . . . 33-2
VRML Viewing . . . . . 33-3
VRML Authoring . . . . . 33-3
Updated Documentation Coming in Printed and Web-Downloadable Form . . . . . 33-3

## 34 Wavelet Toolbox 2.1 Release Notes

**New Features** . . . . . 34-2
disp . . . . . 34-2
wavefun2 . . . . . 34-2
GUI Improvements . . . . . 34-2

**Major Bug Fixes** . . . . . 34-3

**Upgrading from an Earlier Release** . . . . . 34-4

## 35 xPC Target 1.2 Release Notes

**New Features** . . . . . 35-2
C API . . . . . 35-2
UDP Driver Support . . . . . 35-2

Asynchronous Event (Interrupt) Support . . . . . **35-2**
CAN FIFO Support . . . . . **35-3**
100 Mbit/Second Ethernet Support . . . . . **35-3**
Fixed-Point Kernel Support . . . . . **35-3**
Increased Data Upload Speed . . . . . **35-3**
I/O Device Driver Support . . . . . **35-3**

**Known Software and Documentation Problems** . . . . . **35-4**
Uploading Data with External Mode . . . . . **35-4**
Target Command Line Interface . . . . . **35-4**
Memory-Mapped Devices . . . . . **35-4**
Selecting a Working Directory Inside the MATLAB Root Directory . . . . . **35-4**

# Release Notes for Release 12.1

### Upgrading from Earlier Releases

The *Release Notes for Release 12.1* describe differences between Release 12.1 and Release 12.0. These *Release Notes* contain links to the *Release Notes for Release 12*, which provide information about upgrading to Release 12.1 from a release earlier than Release 12.0.

### What's in These Release Notes?

Chapter 1 gives an overview of the major features in Release 12.1. That chapter also summarizes platform limitations and provides some installation notes.

The other chapters describe products with major updates in more detail. As applicable, those chapters include information about a product's:

- New features
- Major bug fixes
- Platform limitations
- Upgrade issues
- Known software and documentation problems

### Core Products with Significant Updates for Release 12.1

- MATLAB
- Simulink
- Stateflow and Stateflow Coder
- Real-Time Workshop

## Other Products with Significant Updates for Release 12.1

| | | |
|---|---|---|
| CDMA Reference Blockset | Financial Time Series Toolbox | Real-Time Embedded Coder |
| Communications Blockset | Fixed-Point Blockset | Real-Time Workshop Ada Coder |
| Control System Toolbox | Image Processing Toolbox | Requirements Management Interface |
| Data Acquisition Toolbox | Instrument Control Toolbox | Signal Processing Toolbox |
| Database Toolbox | MATLAB C/C++ Math Library | Simulink Performance Tools |
| Datafeed Toolbox | MATLAB Runtime Server | Simulink Report Generator |
| DSP Blockset | Neural Network Toolbox | System Identification Toolbox |
| Excel Link | Optimization Toolbox | Wavelet Toolbox |
| Filter Design Toolbox | Power System Blockset | xPC Target |

## New Products for Release 12.1

- Virtual Reality Toolbox
- Developer's Kit for Texas Instruments™ DSP

### Products with Minor Updates for Release 12.1

These products have minor updates for Release 12.1, but are *not* documented in these *Release Notes*. The product names with links take you to the *Release Notes for Release 12* or the *Release Notes for Release 11.1*.

| | | |
|---|---|---|
| Communications Toolbox | MATLAB C/C++ Graphics Library | Nonlinear Control Design Toolbox |
| Dials & Gauges Blockset | MATLAB Report Generator | Partial Differential Equation Toolbox |
| Financial Toolbox | MATLAB Web Server | Real-Time Windows Target |
| Fuzzy Logic Toolbox | Model Predictive Control Toolbox | Robust Control Toolbox |
| MATLAB Compiler | | |

### Printable Version of the Release Notes

A printable version of the *Release Notes for Release 12.1* is available in PDF format.

1

# Introducing Release 12.1

**What's New in Release 12.1?** . . . . . 1-2
Updated Products . . . . . 1-2
New Products . . . . . 1-16

**Platform Limitations** . . . . . 1-17
Differences for the HP and IBM Platforms . . . . . 1-17
Problems Entering Accented Characters on Linux . . . . . 1-18
Problems Entering Accented Characters on UNIX Platforms . 1-18

**Installation Notes** . . . . . 1-19
Maintaining Preference Files After Uninstalling . . . . . 1-19
Correcting the Documentation Location . . . . . 1-19

**Linux Distribution Notes** . . . . . 1-20

**For More Information About What's New in Release 12.1** . . . . . 1-21
Printable Version of the Release Notes . . . . . 1-21

# What's New in Release 12.1?

Release 12.1 includes significant enhancements to our code generation capabilities and the introduction of two new products. This release also includes major updates to the Image Processing Toolbox and several other products, as well as important bug fixes for many products.

This section includes the following two subsections:


- "Updated Products" on page 1-2
- "New Products" on page 1-16


## Updated Products

The table below lists the products that were updated for Release 12.1. For each product, the table indicates whether the update includes new features, bug fixes, or both. The new features for each product are briefly summarized following the table below, and discussed in more detail in a separate chapter in the online version of the *Release Notes for Release 12.1*. Some products include major bug fixes in this release and others include minor bug fixes. The minor bug fixes are not documented in these *Release Notes*.

| Product | New Features | Bug Fixes |
|---|---|---|
| MATLAB® 6.1 | ✔ | major |
| Simulink® 4.1 | ✔ | major |
| Stateflow® 4.1 and Stateflow Coder 4.1 | ✔ | major |
| Real-Time Workshop® 4.1 | ✔ | major |
| CDMA Reference Blockset 1.1 | ✔ | minor |
| Communications Blockset 2.0.1 | ✔ | minor |
| Communications Toolbox 2.0.1 (no chapter in these *Release Notes*) | | minor |
| Control System Toolbox 5.1 | ✔ | minor |

| Product | New Features | Bug Fixes |
|---|---|---|
| Data Acquisition Toolbox 2.1 | ✔ | minor |
| Database Toolbox 2.2 | ✔ | minor |
| Datafeed Toolbox 1.3 | ✔ | minor |
| Developer's Kit for Texas Instruments™ DSP 1.1 | ✔ | minor |
| Dials & Gauges Blockset 1.1.1 (no chapter in these *Release Notes*) | | minor |
| DSP Blockset 4.1 | ✔ | minor |
| Excel Link 1.1.3 | ✔ | minor |
| Filter Design Toolbox 2.1 | ✔ | major |
| Financial Time Series Toolbox 1.1 | ✔ | minor |
| Financial Toolbox 2.1.3 (no chapter in these *Release Notes*) | | minor |
| Fixed-Point Blockset 3.1 | ✔ | minor |
| Fuzzy Logic Toolbox 2.1.1 (no chapter in these *Release Notes*) | | minor |
| Image Processing Toolbox 3.1 | ✔ | minor |
| Instrument Control Toolbox 1.1 | ✔ | minor |
| MATLAB C/C++ Math Library 2.2 | ✔ | minor |
| MATLAB Compiler 2.2 (no chapter in these *Release Notes*) | | minor |
| MATLAB C/C++ Graphics Library 2.2 (no chapter in these *Release Notes*) | | minor |
| MATLAB Report Generator 1.2 (no chapter in these *Release Notes*) | | minor |

| Product | New Features | Bug Fixes |
|---|---|---|
| MATLAB Runtime Server 6.1 | ✔ | minor |
| MATLAB Web Server 1.2.1 (no chapter in these *Release Notes*) | | minor |
| Model Predictive Control Toolbox 1.0.6 (no chapter in these *Release Notes*) | | minor |
| Neural Network Toolbox 4.0.1 | | major |
| Nonlinear Control Design Toolbox 1.1.5 (no chapter in these *Release Notes*) | | minor |
| Optimization Toolbox 2.1.1 | ✔ | minor |
| Partial Differential Equation Toolbox 1.0.4 (no chapter in these *Release Notes*) | | minor |
| Power System Blockset 2.2 | ✔ | minor |
| Real-Time Windows Target 2.1 (no chapter in these *Release Notes*) | | minor |
| Real-Time Workshop Ada Coder 4.1 | ✔ | minor |
| Real-Time Workshop Embedded Coder 2.0 | ✔ | minor |
| Requirements Management Interface 1.0.2 | ✔ | minor |
| Robust Control Toolbox 2.0.8 (no chapter in these *Release Notes*) | | minor |
| Signal Processing Toolbox 5.1 | ✔ | minor |
| Simulink Performance Tools 1.1 | ✔ | minor |
| Simulink Report Generator 1.2 | ✔ | minor |

| Product | New Features | Bug Fixes |
|---|---|---|
| System Identification Toolbox 5.0.1 | ✔ | minor |
| Wavelet Toolbox 2.1 | ✔ | minor |
| xPC Target 1.2 | ✔ | minor |

### MATLAB 6.1

Major enhancements to MATLAB 6.1 involve the following aspects of MATLAB:

- Development environment, including:
  - Editor/Debugger additional functionality and improvements
  - Command Window line wrapping and ability to open the Editor from an error message
  - Improved Workspace Browser sorting
  - Improved source control feature support for Merant PVCS
- Mathematics, including:
  - Evaluation of solutions to differential equation problems
  - Additional functions use Qhull
  - Several new or enhanced math functions
- Programming and data types, including:
  - Support for partial evaluation of `if` and `while` expressions
  - A new MATLAB search string function (`strfind`)
  - New file I/O functions for scientific data formats
  - New audio functions
- Graphics, including:
  - Additional support for Ghostscript
  - Transparent legends
- OpenGL renderer support

- External interfaces, including:
  - Concatenation of Java arrays
  - New Fortran MEX, MX, MAT, and ENG functions
  - Serial I/O enhancements
- Creating graphical user interfaces (GUIDE), with enhancements to the Layout Editor and the Menu Editor

### Simulink 4.1

Major enhancements to Simulink 4.1 include:

- The Simulink Editor new features and enhancements, such as:
  - Autoconnection and autorouting for blocks and signals
  - Undo option improvements
  - Ability to save Release 11 format models
- Modeling enhancements, including an S-Function Builder for generating an S-function from a specification and generating code for Real-Time Workshop
- Block Library additions and enhancements, such as:
  - New control flow blocks For Iterator, If, Switch Case, and While Iterator
  - New Bus Creator block for combining the output of multiple blocks into a single signal bus
  - Additional data type support for some blocks
  - Floating Scope block, with enhancements for signal selection and saving selected signals

---

**Note** See “Simulink Performance Tools 1.1” on page 1-14, if you have that product.

---

### Stateflow 4.1 and Stateflow Coder 4.1

The major enhancements to Stateflow 4.1 and Stateflow Coder 4.1 include:

- More search capabilities added to Stateflow’s Search & Replace tool
- Stateflow charts now include the graphical innovation of “smart” transitions whose ends slide around the surfaces of states and junctions.

- Chart annotation that is similar to Simulink annotation
- The Simulink Model Coverage tool now performs model coverage calculations for decisions and conditions of decision

### Real-Time Workshop 4.1

Real-Time Workshop 4.1 has significant new and enhanced features and many improvements in the quality of generated code, including:

- Expression folding, which removes temporary variables, making the code more efficient and readable
- External mode support for inlined parameters
- Block states interface to externally written code, in a manner similar to signals.
- New debugger for Target Language Compiler (TLC) programs
- Support for new Simulink blocks, including control flow constructs such as `do-while`, `for`, and `if`
- Numerous bug fixes

---

**Note** See "Real-Time Workshop Ada Coder 4.1" on page 1-13 and "Real-Time Embedded Coder 2.0" on page 1-13, if you have those products.

---

### CDMA Reference Blockset 1.1

The CDMA Reference Blockset 1.1 supports frame-based signals and matrix signals. As a result, it is no longer necessary to change the signal attributes of blocks from the Communications Blockset or the DSP Blockset to use them in models with blocks from the CDMA Reference Blockset.

### Communications Blockset 2.0.1

The enhancements to the Communications Blockset 2.0.1 include:

- A script that sets certain model preferences appropriately
- New blocks that convert between bipolar and unipolar signals
- A new function that generates seeds for random-output blocks

- The ability to have error statistics determine the duration of the simulation
- A more flexible integrator block

The Communications Blockset 2.0.1 online documentation set now includes a new document, called "Getting Started with the Communications Blockset."

### Control System Toolbox 5.1

Major enhancements to the Control System Toolbox 5.1 include:

- Several important enhancements to the SISO Design Tool to make it more powerful and easier to use
- Enhancements to controls algorithms, such as:
  - M-circle support in the `nyquist` function
  - A new function, `bandwidth`, that calculates the bandwidth of SISO systems
  - For LTI models with complex data, an option to specify transfer functions or state-space models with complex coefficients

### Data Acquisition Toolbox 2.1

Major enhancements to the Data Acquisition Toolbox 2.1 include support for:

- The following additional National Instruments boards:
  - 445x Dynamic Signal Acquisition (DSA) Series boards
  - Analog output boards
  - Digital I/O boards (no streaming)
- High resolution (greater than 16-bit) sound cards. Note that some sound card drivers may not support this feature because they are still in beta.

### Database Toolbox 2.2

Major enhancements to the Database Toolbox 2.2 include:

- Retrieval of data as numeric, structure, or cell array format, as specified via preferences. In previous versions of the Database Toolbox, only the cell array format was supported. The numeric format allows for a significant reduction in memory usage. The structure format allows the retrieved data to be accessed as a MATLAB structure.

- A preference for handling errors to be `store`, `report`, or `empty`. In previous versions of the Database Toolbox, only the `store` option was supported. The `report` option immediately displays any retrieval error messages in the Command Window.
- Support for `LONGCHAR` data type, which is called the `Memo` data type in Microsoft Access.

### Datafeed Toolbox 1.3

The Datafeed Toolbox 1.3 adds the Hyperfeed data server to the list of previously supported servers (i.e., Bloomberg, FT Interactive Data, and Yahoo).

### Developer's Kit for Texas Instruments DSP 1.1

---

**Note** The Developer's Kit for Texas Instruments™ DSP 1.0 was released in a Web-downloadable format prior to Release 12.1. Version 1.1 of this product includes some important updates to Version 1.0, as summarized below.

---

The major Developer's Kit for Texas Instruments DSP 1.1 enhancements include:

- Five new functions that extend the program debugging and project management capabilities of the Developer's Kit
- New interface between the Developer's Kit and FDATool
- Unified CCS IDE and C6701 EVM targeting
- New run-time libraries
- Architectural changes to the CCSDSP object to speed up processing and support future target development

### DSP Blockset 4.1

Major enhancements to the DSP Blockset 4.1 include:

- The new Digital Filter Design block, which implements filters that you design in the Filter Design and Analysis Tool (FDATool)
- Enhancements to several blocks:
  - Major enhancements to the FFT, IFFT, Magnitude FFT, Overlap-Add FFT, Overlap-Save FFT, Random Source, Sine Wave, Unwrap, and Window Function blocks
  - Additional data type support of Simulink built-in data types, fixed-point data types, and custom data types
  - Improved memory management for both simulation and Real-Time Workshop C code generation
  - Smaller, more efficient Real-Time Workshop generated code
- Extensible C-callable simulation and run-time library functions, macros, and definitions for use with your S-functions and Real-Time Workshop C code

### Excel Link 1.1.3

Major enhancements to the Excel Link 1.1.3 include:

- Installation under control of the FLEXlm license manager, as with other MathWorks products
- Additional error detection

### Filter Design Toolbox 2.1

Major enhancements to the Filter Design Toolbox 2.1 include:

- Many new adaptive filtering functions
- Four new FIR filter design functions
- The new `zerophase` analysis function that returns the zero-phase response of a linear-phase FIR filter
- Several new filter transformation functions
- The FDATool's new Transformation option that can save you from repeating the filter design process
- Context-sensitive help for the FDATool Quantization mode

### Financial Time Series Toolbox 1.1

The Financial Time Series Toolbox 1.1 includes the following features that were introduced in the Web-downloadable release of this product prior to Release 12.1:

- Several new functions, including bar chart plotting functions
- A new Financial Time Series GUI that provides complete access to the full functionality of the Financial Time Series Toolbox

### Fixed-Point Blockset 3.1

Major enhancements to the Fixed-Point Blockset 3.1 include:

- Tools to support approximating functions with look-up tables
- Support for matrix and frame-based signals
- More efficient code generation
- Many new blocks that enable you to do common operations more easily; some of the operations these new blocks support are:
  - Calculations that depend on sample time
  - Generating arbitrary wave forms
  - Limiting the rate of control signals

### Image Processing Toolbox 3.1

Major enhancements to the Image Processing Toolbox 3.1 include those introduced in Version 3.0, which was released in Web-downloadable form shortly before Release 12.1. The new features introduced in the Image Processing Toolbox, Version 3.0, include:

- Many new morphology functions
- New spatial transformation functions
- New image registration functions, with a new graphical user interface
- New integer image arithmetic functions
- New Integer image filtering function
- New image deblurring (deconvolution) functions
- Support for DICOM files

- Miscellaneous new functions
- New image processing demos

The new features introduced in the Image Processing Toolbox, Version 3.1, include:

- New deblurring function, `deconvblind`, that implements the blind deconvolution algorithm
- New utility function, `label2rgb`, that converts a label matrix into an RGB color image

### Instrument Control Toolbox 1.1

The major enhancement to the Instrument Control Toolbox 1.1 is Binary block (binblock) data transfer.

### MATLAB C/C++ Math Library 2.2

Major enhancements to the MATLAB C/C++ Math Library 2.2 include:

- New library functions that evaluate the solution of a differential equation problem using the output of `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`, `ode23t`, `ode23tb`, or `bvp4c`
- Enhancements to library functions for:
  - Eigenvalues
  - Eigenvectors
  - Ordinary differential equations (ODEs)

### MATLAB Runtime Server 6.1

The MATLAB Runtime Server 1.1 includes a new tool, `makeconfig`, for generating the configuration file for packaging an application.

### Optimization Toolbox 2.1.1

In the Optimization Toolbox 2.1.1, large-scale finite differencing is improved numerically to handle cases when an optimization problem is not well scaled.

### Power System Blockset 2.2

Major enhancements to the Power System Blockset 2.2 include:

- The new Powergui tool **Display spectrum of Signal** feature, which displays the fast Fourier transform (FFT) of a specified signal
- Documentation for several blocks in the powerlib_extras library that were previously undocumented

### Real-Time Workshop Ada Coder 4.1

The Real-Time Workshop Ada Coder 4.1 supports all applicable new features added to the Real-Time Workshop 4.1.

### Real-Time Embedded Coder 2.0

Major enhancements to the Real-Time Embedded Coder 2.0 include:

- Significant improvements in efficiency and readability of generated code (many improvements in Real-Time Workshop code generation technology are especially applicable to embedded systems development, such as expression folding and buffer optimizations)
- Custom storage classes for signal, state, and parameter objects, for embedded systems development
- Additional and enhanced code generation options
- Improved HTML code generation report
- Generated ERT S-function, which supports the Fixed-Point Blockset
- More streamlined model initialization

### Requirements Management Interface 1.0.2

The Requirements Management Interface 1.0.2 now works with the latest DOORS release, Version 5.1.

### Signal Processing Toolbox 5.1

Major enhancements to the Signal Processing Toolbox 5.1 include:

- The new Filter Visualization Tool (`fvtool`) that groups all the filter analysis functions, such as magnitude, phase, impulse response, etc., into a single window where you can easily switch between analyses, add annotations, and print your filter response

- Many FDATool enhancements
- The ability to print preview and print your filter design in the SPTool Filter Viewer and Filter Designer
- Six new window functions and a general window function
- Five other new functions and enhancements to several functions
- A new demo, Vector Transition Processing Using Spectral Windows

### Simulink Performance Tools 1.1

Major enhancements to the Simulink Performance Tools 1.1 to two tools:

- Model Coverage Tool improvements, including:
  - Coverage analysis extended to include coverage for conditions (Logic block and Stateflow transitions), modified condition/decisions (MC/DC), and look-up tables.
  - Increased coverage for more blocks
- Renaming the Model Differencing Tool to be the Graphical Merge Tool, to reflect its new model merging capability, including:
  - Options to compare in-memory, on-disk, and source control system versions of models
  - Improvements to graphical user interface display

### Simulink Report Generator 1.2

Major enhancements to the Simulink Report Generator 1.2 include:

- The Block Type: Look-Up Table component that now reports on N-D look-up tables (as well as 1-D and 2-D look-up tables)
- For some components, removal of the previous requirement for Real-Time Workshop
- The ability to add titles to images for the System Snapshot and Stateflow Snapshot components

### System Identification Toolbox 5.0.1

The System Identification Toolbox 5.0.1 adds a new function, `getexp`, which obsoletes the `data(...)` syntax.

### Wavelet Toolbox 2.1

Major enhancements to the Wavelet Toolbox 2.1 include:

- Reorganization and improvement of GUI menus
- The new `disp` method, which displays information for a WPTREE object
- The new `wavefun2` function returns the 2-D scaling and wavelet functions from tensor products of 1-D scaling and wavelet functions

### xPC Target 1.2

Major enhancements to the xPC Target 1.2 include:

- A C API to create stand-alone graphical user interfaces (GUIs) with a real-time target application (xPC Target includes examples for Visual C/C++ and Visual BASIC)
- Support for:
  - User Datagram Protocol (UDP) packets
  - CAN FIFO model I/IO drivers, for monitoring an entire CAN network or minimizing driver latency for certain applications
  - Additional I/O device drivers
  - 10 or 100 megabit/second communication between the host and target computers for downloading real-time application and uploading signal data
  - Fixed-point kernel, for PCs (CPUs) that do not have a hardware floating-point unit
  - The creation of subsystems that execute asynchronously when an event (interrupt) occurs (these subsystems can communicate and be part of a larger model that executes with a monotonic step size)
- Log data uploading about five times faster or better; scope data uploading about three times faster

## New Products

The following products are new since Release 12.0. These products were made available in Web-downloadable form prior to the availability of Release 12.1:

- Virtual Reality Toolbox 2.0
- Developer's Kit for Texas Instruments™ DSP 1.1

---

**Note** The Developer's Kit for Texas Instruments™ DSP 1.0 was released as in a Web-downloadable format prior to Release 12.1. Version 1.1 of this product includes some important updates to Version 1.0.

---

# Platform Limitations

For Release 12.1, all the features described in these *Release Notes* and in the Release 12.1 documentation are supported in *full* for the following platforms:

- Microsoft Windows 95, Windows 98, Windows 2000, Windows Millenium, and Windows NT
- Compaq Alpha
- Linux
- SGI
- Sun Solaris

---

**Note** As in Release 11 and Release 12, there are some minor differences between Windows and UNIX for specific tools, as documented for those tools.

---

For details about each of these platforms, see the Support page of the MathWorks Web page, at `http://www.mathworks.com`.

## Differences for the HP and IBM Platforms

---

**Note** The Release 12 platform limitations for the HP and IBM platforms still apply to Release 12.1. In addition, there are a few limitations new in Release 12.1, as described below.

---

**Release 12.1-Specific Limitations on HP and IBM Platforms**

| Product | Limitations on HP and IBM Platforms |
|---|---|
| Image Processing Toolbox | The `cpselect` GUI is not supported. |
| Simulink | The following new Simulink features are *not* supported:<br>• Simulink Data Class Designer<br>• S-Function Builder |

## Problems Entering Accented Characters on Linux

On Linux platforms, using an accent key to accent a character in a two-stroke sequence in a MATLAB window, such as ^e for ê or ~n for ñ, may cause MATLAB to hang. To avoid this problem in MATLAB, use `matlab -nodesktop`.

With Simulink, avoid the use of these keystrokes. To use characters such as the caret ^ when it is mapped as a dead-accent key, you must paste it into the MATLAB IDE from a non-MATLAB window, such as an xterm, or use the `power` and `mpower` functions directly.

## Problems Entering Accented Characters on UNIX Platforms

Keys for accented characters, such as Ü on German keyboards and Å on Swedish keyboards are ignored in the IDE. Pasting these characters into MATLAB windows results in the character value 128 instead of the actual character value. For example, pasting the string `double( ¨˚ )` into MATLAB appears as

```
double(' ')
ans =
128   128   128   128   128   128   128
```

# Installation Notes

## Maintaining Preference Files After Uninstalling

If you use the Release 12 uninstaller, it will automatically remove your preference files including the Command History, preferences, and Help browser favorites.

To keep these files for use in Release 12.1, make a copy of them before uninstalling Release 12. To see where the files are located, run `prefdir` in the Command Window. The relevant files are listed below.

| Filename | File For |
|---|---|
| `cwdhistory.m` | Command Window history |
| `history.m` | Command History |
| `matlab.prf` | Preferences |
| `matlab_help.hst` | Help browser favorites |

After uninstalling Release 12, put your backup copy of the files in the location returned by `prefdir` so that Release 12.1 can use the files.

## Correcting the Documentation Location

On Windows concurrent license systems and on UNIX systems, if you installed Release 12.1, including the help files, in a different location than Release 12.0 (as is recommended), the Help browser will not find the Release 12.1 help files. When you run the Help browser, you will either see the Release 12.0 help files or you will see a warning stating that the help files cannot be found. To use the Release 12.1 help files, select **File -> Preferences -> Help** and change the **Documentation Location** for **Local or network directory** to the location of Release 12.1 help files. The location is the `help` directory under the MATLAB Release 12.1 directory, for example, `d:\matlab_r12_1\help`.

For single user Windows systems, the Help browser should automatically find the Release 12.1 help files, but if it does not, change the **Documentation Location** as described above.

## Linux Distribution Notes

As of this writing, the following distributions are known to work:

- Redhat 7.0
- Mandrake 8.n
- SuSE 7.0
- Debian 2.1 and 2.2
- Slackware 7.1

# For More Information About What's New in Release 12.1

For more detailed information about each product that is included in Release 12.1, see the sections of the online version of the *Release Notes* for the product(s) of interest to you. In those sections, as applicable, you can find out about a product's:

- New features
- Major bug fixes
- Platform limitations
- Issues relating to upgrading from an earlier release
- Known software and documentation problems

The online *Release Notes* include links to the rest of the online documentation, where you can get detailed information about how to use the new and enhanced features.

## Printable Version of the Release Notes

A printable version of the *Release Notes for Release 12.1* is available in PDF format.

# MATLAB Software Acknowledgments

MATLAB and its associated products incorporate the following third-party software:

ARnoldi PACKage (ARPACK)

Rich Lehoucq, Kristi Maschhoff, Danny Sorensen, and Chao Yang
`http://www.caam.rice.edu/software/ARPACK`

Automatically Tuned Linear Algebra Software (ATLAS)

R. Clint Whaley and Jack Dongarra
`http://www.netlib.org/atlas`

`fft` and related MATLAB functions are based on the FFTW library

Developed by Matteo Frigo and Steven G. Johnson
Copyright © 1997-1999 Massachusetts Institute of Technology. All rights reserved.
Used under terms of a commercial license
`http://www.fftw.org`

HDF capability in the functions `imread`, `imwrite`, `imfinfo`, and `hdf` is based on code of which portions were developed at

The National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign.

JPEG capability in the functions `imread`, `imwrite`, `imfinfo`, `print`, and `saveas`

This software is based in part on the work of the Independent JPEG Group.

Linear Algebra PACKage (LAPACK)

`http://www.netlib.org/lapack` (for general information about LAPACK)
For details, see the *LAPACK User's Guide*.
E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen
For a printed version of the *LAPACK User's Guide*, go to `http://www.siam.org`.
For an online version of the *LAPACK User's Guide*, go to
`http://www.netlib.org/lapack/lug/lapack_lug.html`.

Qhull based computational geometry capability in MATLAB

Qhull copyright (c) 1993 The National Science and Technology Research Center for Computation and Visualization of Geometric Structures, The Geometry Center, University of Minnesota
e-mail: `software@geom.umn.edu`
For complete copyright information, issue the MATLAB command `help qhull`.

Sparse matrix minimum degree permutation functions `colamd` and `symamd`

Copyright © 1998-2000 by the University of Florida. All rights reserved.
Authors of the code are Stefan I. Larimore and Timothy A. Davis (`davis@cise.ufl.edu`), University of Florida. The algorithm was developed in collaboration with John Gilbert, Xerox PARC, and Esmond Ng, Oak Ridge National Laboratory.
This work was supported by the National Science Foundation, under grants DMS-9504974 and DMS-9803599.
For complete copyright information, issue the MATLAB command `edit colamd` or `edit symamd`.

TIFF capability in the functions `imread`, `imwrite`, `imfinfo`, `print`, and `saveas`:

Copyright © 1988-1999 Sam Leffler

Copyright © 1991-1999 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

2

# MATLAB 6.1 Release Notes


**New Features** . . . . . 2-2
Development Environment Features . . . . . 2-2
Mathematics Features . . . . . 2-5
Programming and Data Types Features . . . . . 2-8
Graphics Features . . . . . 2-10
OpenGL Renderer Feature — Microsoft Windows . . . . . 2-11
External Interfaces/API Features . . . . . 2-12
Creating Graphical User Interfaces — GUIDE . . . . . 2-17

**Major Bug Fixes** . . . . . 2-18
Development Environment . . . . . 2-18
Mathematics . . . . . 2-18

**Platform Limitations** . . . . . 2-20
Patch Required for HP-UX 11.0 . . . . . 2-20
Mathematics Limitations . . . . . 2-20
Graphics and Printing Limitations . . . . . 2-21
GUIDE Limitations . . . . . 2-21
You May Need to Overwrite the MATLAB Default Choice of BLAS . . . . . 2-22

**Upgrading from an Earlier Release** . . . . . 2-26
Development Environment Issues . . . . . 2-26
Mathematics Issues . . . . . 2-27
Programming and Data Types Issues . . . . . 2-28
Graphics Issue . . . . . 2-29
External Interfaces/API Issues . . . . . 2-30

**Known Software and Documentation Problems** . . . . 2-32
Development Environment Problems . . . . . 2-32
Documentation Updates . . . . . 2-33

# New Features

This section introduces the new features and enhancements added in MATLAB 6.1 since MATLAB 6.0 (Release 12.0).

For information about MATLAB features that are incorporated from recent releases, see the *Release Notes for Release 12*.

This section about new features is organized into the following subsections:


- "Development Environment Features" on page 2-2
- "Mathematics Features" on page 2-5
- "Programming and Data Types Features" on page 2-8
- "Graphics Features" on page 2-10
- "OpenGL Renderer Feature — Microsoft Windows" on page 2-11
- "External Interfaces/API Features" on page 2-12
- "Creating Graphical User Interfaces — GUIDE" on page 2-17


## Development Environment Features

### Command Window

MATLAB 6.1 includes two command window enhancements:

- You can set a preference for the command window to wrap lines. Input and output lines wrap to fit within the current width of the command window.
- If an error message appears when running an M-file, click on the underlined portion of the error message, or press **Ctrl+Enter**. The offending M-file opens in the Editor, scrolled to the line containing the error.

### Help Browser

When you select documentation for the product filter, you can clear all currently selected products or select all products.

### Editor / Debugger

The Editor/Debugger has the following enhancements:

- You can set bookmarks in M-files in the Editor/Debugger so that you can go directly to a particular line in the file. To set a bookmark, position the cursor at the line you want to bookmark, and then select **Set/Clear Bookmark** from the **Edit** menu.

  After setting bookmarks, you can go to the next or previous bookmark in a file. This allows you to go directly to a marked spot. Use the **Edit** menu items **Next Bookmark** and **Previous Bookmark** to navigate. Bookmarks are not saved when you close a file.
- You can include line numbers when printing files from the Editor/Debugger. To include line numbers, select **Preferences -> Editor/Debugger -> Printing**. Under **Print options**, check **Print line numbers**.
- You can use keyboard shortcuts to comment or uncomment a selection in the Editor/Debugger. The shortcuts are platform dependent and are listed with the menu items on the Editor/Debugger **Text** menu.
- In the **Find/Replace** dialog box, settings for **Match case**, **Whole word**, and **Wrap around** are remembered for the next MATLAB session.
- You can find the previous occurrence of a selection in the Editor/Debugger by pressing **Ctrl+Shift+F3**. You can also find the previous occurrence of a string you entered into the **Find & Replace** dialog box by pressing **Shift+F3**.
- When you move an arrow key over a token, for example, an opening parenthesis, (, the token and its match are briefly underlined. If there is no matching token, the token appears with a strike-through mark, ( with a strike-through ).
- When you run a file from the Editor/Debugger and the file is not in a directory on the search path or in the current directory, a dialog box appears presenting you with options that allow you to run the file. You can either change the current directory to the directory containing the file, or you can add to the search path the directory containing the file.

  If the file you want to run is already in a directory on the search path or in the current directory, the current directory remains as is and there are no actions you need to take.

- When you add a breakpoint to a file that is not in a directory on the search path or in the current directory, a dialog box appears presenting you with options that allow you to add the breakpoint. You can either change the current directory to the directory containing the file, or you can add to the search path the directory containing the file.

  If the file you want to run is already in a directory on the search path or in the current directory, the current directory remains as is and there are no actions you need to take.
- If you type `edit filename` and `filename` does not exist, a prompt appears asking if you want to create a new file. If you select **Yes**, a blank file titled `filename.m` is created in the Editor/Debugger. You can turn off this option in preferences for the Editor/Debugger.

### Current Directory Browser

In the **Find/Replace** dialog box, settings for **Match case**, **Whole word**, and **Subdirectories** are remembered for the next MATLAB session.

Also, you can delete directories that are not empty. All contents of the directory will be deleted along with the directory.

### Workspace Browser

You can select the column on which to sort in the Workspace browser, as well as reverse the sort order of any column. Click on a column heading to sort on that column. Click on the column heading again to reverse the sort order in that column. For example, to sort on **Size**, click the column heading once. To change from ascending to descending, click on the heading again.

### Source Control

If you use Merant PVCS with MATLAB source control features, you no longer need to specify the project configuration file using `cmopts`. If you did specify it in previous releases, you do not have to remove it as MATLAB will ignore it.

### General

The `computer` function now displays the endian byte ordering of the computer with the following form.

```
[str,maxsize,endian] = computer
```

# Mathematics Features

## Evaluation of Solutions to Differential Equation Problems

A new function, `deval`, enables you to evaluate the solution of a differential equation problem at a vector of points from the interval in which the problem was solved. `deval` uses, as input, the output structure `sol` of an initial value problem solver (`ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`, `ode23t`, `ode23tb`) or the boundary value problem solver (`bvp4c`). A new ODE solver syntax returns the structure `sol`.

## Additional Functions Use Qhull

These functions are now based on Qhull:

- `delaunay` — two-dimensional Delaunay triangulation
- `convhull` — two-dimensional convex hull

These functions call `delaunay` and therefore are now indirectly based on Qhull:

- `voronoi` — two-dimensional Voronoi diagrams
- `griddata` — data gridding and surface fitting

These functions are in addition to the Qhull-based functions introduced in MATLAB 6.0 (Release 12.0): `convhulln`, `delaunay3`, `delaunayn`, `griddata3`, `griddatan`, and `voronoin`.

## Math Function Summary Tables

This section summarizes:

- New math functions
- Functions with new or changed capabilities

---

**Note** See “Upgrading from an Earlier Release” on page 2-26 for information about obsolete functions.

---

### New Math Functions

| Function | Purpose |
|---|---|
| deval | Evaluate the solution of a differential equation problem using the output of ode45, ode23, ode113, ode15s, ode23s, ode23t, ode23tb, or bvp4c. |
| erfcinv | Inverse complementary error function. |
| tetramesh | Tetrahedron mesh plot for use with delaunayn. |
| triplot | 2-D triangular plot for use with delaunay. |

### Math Functions with New or Changed Capabilities

| Function | Enhancement/Change |
|---|---|
| bvpinit | New syntax solinit = bvpinit(sol,[anew bnew]) extrapolates a solution sol as an initial guess for solving a BVP on an extended interval. It can copy parameters from the previous iteration or let the user to provide new ones. For more information, see "Boundary Value Problems for ODEs" in the MATLAB documentation. |
| bvpset | New Vectorized option lets you pass to the solver bvp4c an array of column vectors. This allows bvp4c to reduce the number of function evaluations, and may significantly reduce solution time. For more information see "Boundary Value Problems for ODEs" in the MATLAB documentation. |
| convhull | New syntax [K,a] = convhull(x,y) returns the area a of the convex hull. |
| convhulln | New syntax [K,v] = convhulln(X) returns the volume v of the convex hull. |

**Math Functions with New or Changed Capabilities (Continued)**

| Function | Enhancement/Change |
|---|---|
| `numel` | New syntax `n = numel(A, varargin)` returns the number of subscripted elements, `n`, in `A(index1,index2,...,indexn)`, where `varargin` is a cell array whose elements are `index1`, `index2`, `...`, `indexn`. |
| `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`, `ode23t`, `ode23tb` | New syntax `sol = solver(odefun,[t0 tf],y0...)` returns a structure that you can use with the new function `deval` to evaluate the solution at any point on the interval `[t0,tf]`. |
| `polyeig` | New syntax `e = polyeig(A0,A1,..,Ap)` returns only the eigenvalues of the specified eigenvalue problem. Use `[X,e] = polyeig(A0,A1,...Ap)` if you also want the eigenvectors. This capability is available in MATLAB 6.0 (Release 12.0). |
| `ppval` | New syntax `ppval(xx,pp)` transposes the input arguments to enable you to use `ppval` with function functions. |
| `qz` | New syntax `[AA,BB,Q,Z,V,W] = qz(A,B)` returns `W`, the left generalized eigenvectors of `A` and `B`. |
| `reshape` | New syntax `reshape(A,...,[],...)` calculates the length of the dimension specified by the placeholder `[]`. |
| `svd` | Can now return only the first two outputs, `U` and `S`, where `S` is a diagonal matrix of the same dimension as the input argument `X`, and `U` is a unitary matrix. |

# Programming and Data Types Features

## Partial Evaluation of Expressions

Within the context of an `if` or `while` expression, MATLAB does not necessarily evaluate all parts of a logical expression. In some cases, it is possible, and often advantageous, to determine whether an expression is true or false through only partial evaluation. This is sometimes referred to as *short-circuiting*.

For example, if `A` equals zero in statement 1 below, then the expression evaluates to `false`, regardless of the value of `B`. In this case, there is no need to evaluate `B` and MATLAB does not do so. In statement 2, if `A` is nonzero, then the expression is `true`, regardless of `B`. Again, MATLAB does not evaluate the latter part of the expression.

```
1)  if (A & B)        2)  if (A | B)
```

You can use this property to your advantage to cause MATLAB to evaluate a part of an expression only if a preceding part evaluates to the desired state.

---

**Note** Partial evaluation of expressions in `if` and `while` was also available in MATLAB 6.0, but was not documented.

---

## New MATLAB Search String Function

`strfind` is a new character array function in MATLAB. It searches for all occurrences of a string pattern within another, longer string. Placement of the two string arguments in the argument list requires that you be specific about which string is the character pattern to search for and which is the string in which to search. This allows you more control over how the search is performed compared with the MATLAB `findstr` function, particularly when executing searches within a loop.

### New File I/O Functions for Scientific Data Formats

There are six new MATLAB 6.1 functions that enable you to retrieve information and data from Common Data Format (CDF), Flexible Image Transport System (FITS), and Hierarchical Data Format (HDF) files.

| Function | Purpose |
|---|---|
| `cdfinfo` | Return information about a CDF file |
| `cdfread` | Read data from a CDF file |
| `fitsinfo` | Return information about a FITS file |
| `fitsread` | Read data from a CDF file |
| `hdfinfo` | Return information about an HDF or HDF-EOS file |
| `hdfread` | Read data from an HDF or HDF-EOS file |

### New Audio Functions

MATLAB 6.1 includes two new audio functions for 32-bit Windows platforms only.

| Function | Purpose |
|---|---|
| `audioplayer` | Create an audio object to play audio data |
| `audiorecorder` | Create an audio object to record audio data |

### Date Conversion Changes

The `datenum` and `datestr` functions can now accept a date vector, as defined by `datevec`, as an input argument. For example, `datestr(clock)` returns the current date and time as string such as `27-Apr-2001 15:58:41`.

## Graphics Features

### Transparent Legend

You can now make the legend box transparent, enabling you to see the plotted data behind the legend. See `legend` for more information.

### New Ghostscript Drivers

The following new Ghostscript drivers are available with MATLAB by using the device switch shown below.

| Printer Driver | Device Switch |
|---|---|
| Canon Color BubbleJet BJC-800 | `-dbjc800` |
| HP LaserJet 4.5L and 5P | `-dljet4` |
| HP LaserJet 5 and 6 | `-dpxlmono` |

### New Ghostscript Output Filters for Exporting

The following new Ghostscript output filters are available with MATLAB by using the option switch shown below.

| File Format | Option Switch |
|---|---|
| BMP Monochrome BMP | `-dbmpmono` |
| PDF Color file Format | `-dpdf` |

### Higher Resolution Metafiles

You can now set the resolution of a Windows Enhanced Metafile copied from a MATLAB figure window with the `print -dmeta` command. Set the resolution using the `-d` option of the `print` command. For example, to copy a figure to a metafile having a resolution of 200 dpi, use

```
print -dmeta -r200
```

MATLAB uses the screen resolution as the default.

### Default PaperType and PaperUnits Set For International Users

The `matlabrc.m` startup file now sets the default `PaperType` and `PaperUnits` properties based on ISO Country Codes. These default to `'a4'` and `'centimeters'` respectively for users in countries that normally default to these settings. Other countries still default to `'usletter'` and `'inches'`.

The same values are used for default Simulink `PaperType` and `PaperUnits` properties in the `matlabrc.m` startup file.

You can still set default `PaperType` or `PaperUnits` values yourself by adding the following to `startup.m`.

```
set(0, 'DefaultFigurePaperType', 'a4')
set(0, 'DefaultFigurePaperUnits', 'centimeters')
```

## OpenGL Renderer Feature — Microsoft Windows

If you do not want to use hardware OpenGL, but do want to use object transparency, you can issue the following command.

```
feature('UseGenericOpenGL',1)
```

This command forces MATLAB to use generic OpenGL on Microsoft Windows platforms. Generic OpenGL is useful if your hardware version of OpenGL does not function correctly and you want to use image, patch, or surface transparency, which requires the OpenGL renderer.

To re-enable hardware OpenGL, use the command

```
feature('UseGenericOpenGL',0)
```

Note that the default setting is to use hardware OpenGL

To query the current state of the generic OpenGL feature, use the command

```
feature('UseGenericOpenGL')
```

See the `opengl` reference page for additional information.

# External Interfaces/API Features

## Concatenation of Java Arrays

In MATLAB 6.1, you can concatenate arrays of Java objects that have unlike dimensions. The following example concatenates a 2-by-3 array of `java.lang.Integer` with a 4-by-3 array of the same class.

```
A =
java.lang.Integer[][]:
   [ 1]    [ 2]    [ 3]
   [ 4]    [ 5]    [ 6]
   [17]    [18]    [19]
   [20]    [21]    [22]

B =
java.lang.Integer[][]:
   [11]    [12]    [13]
   [14]    [15]    [16]
```

The vertical concatenation `[A;B]` is simple since both arrays have the same number of columns. The horizontal concatenation `[A B]` merges the two arrays into an irregularly shaped array having six columns in the first and second rows and three columns in the third and fourth rows.

```
C = [A;B]
C =
java.lang.Integer[][]:
    [ 1]    [ 2]    [ 3]
    [ 4]    [ 5]    [ 6]
    [11]    [12]    [13]
    [14]    [15]    [16]
    [17]    [18]    [19]
    [20]    [21]    [22]
```

```
C = [A B]
C =
java.lang.Integer[][]:
   [6 element array]
   [6 element array]
   [3 element array]
   [3 element array]
```

---

**Note** "Concatenation of Java Objects" on page 2-30 discusses changes to how Java objects are concatenated.

---

### New Fortran MX, MEX, MAT, and ENG Functions

The following functions have been added to the Fortran MX, MEX, MAT, and Engine external interface. Most of these functions already exist in the MATLAB C language API.

**Table 2-1: New Fortran MX Functions**

| | |
|---|---|
| mxAddField | mxCalcSingleSubscript |
| mxClassIDFromClassName | mxClearLogical |
| mxCopyComplex8ToPtr | mxCopyInteger1ToPtr |
| mxCopyInteger2ToPtr | mxCopyPtrToComplex8 |
| mxCopyPtrToInteger1 | mxCopyPtrToInteger2 |
| mxCopyPtrToReal4 | mxCopyReal4ToPtr |
| mxCreateCellArray | mxCreateCellMatrix |
| mxCreateCharArray | mxCreateCharMatrixFromStrings |
| mxCreateDoubleMatrix | mxCreateNumericArray |
| mxCreateNumericMatrix | mxCreateScalarDouble |
| mxCreateStructArray | mxCreateStructMatrix |
| mxDestroyArray | mxDuplicateArray |
| mxGetCell | mxGetClassID |
| mxGetClassName | mxGetData |
| mxGetDimensions | mxGetElementSize |
| mxGetEps | mxGetField |
| mxGetFieldByNumber | mxGetFieldNameByNumber |
| mxGetFieldNumber | mxGetImagData |
| mxGetInf | mxGetNaN |
| mxGetNumberOfDimensions | mxGetNumberOfElements |
| mxGetNumberOfFields | mxIsCell |

**Table 2-1: New Fortran MX Functions (Continued)**

| | |
|---|---|
| mxIsChar | mxIsClass |
| mxIsEmpty | mxIsFinite |
| mxIsFromGlobalWS | mxIsInf |
| mxIsInt8 | mxIsInt16 |
| mxIsInt32 | mxIsLogical |
| mxIsNaN | mxIsSingle |
| mxIsStruct | mxIsUint8 |
| mxIsUint16 | mxIsUint32 |
| mxMalloc | mxRealloc |
| mxRemoveField | mxSetCell |
| mxSetData | mxSetDimensions |
| mxSetField | mxSetFieldByNumber |
| mxSetImagData | mxSetLogical |

**Table 2-2: New Fortran MEX Functions**

| | |
|---|---|
| mexFunctionName | mexGetArray |
| mexGetArrayPtr | mexIsGlobal |
| mexIsLocked | mexLock |
| mexMakeArrayPersistant | mexMakeMemoryPersistant |
| mexPutArray | mexUnlock |
| mexWarnMsgTxt | |

**Table 2-3: New Fortran MAT Functions**

| | |
|---|---|
| matDeleteArray | matGetArray |
| matGetArrayHeader | matGetNextArray |

**Table 2-3: New Fortran MAT Functions (Continued)**

| | |
|---|---|
| matGetNextArrayHeader | matPutArray |
| matPutArrayAsGlobal | |

**Table 2-4: New Fortran Engine Functions**

| | |
|---|---|
| engGetArray | engPutArray |

### Property Added to ActiveX and Engine Interfaces

For ActiveX automation server applications and MATLAB Engine applications running on Windows, you can control whether the application windows appear on the Windows desktop with a new property called `Visible`.

When `Visible` is set, the ActiveX application or engine server window is visible on the desktop, thus enabling user interaction with the server. This is the default. When `Visible` is cleared, the application or engine window is removed from the desktop.

**ActiveX.** This example disables the visibility of an ActiveX automation server application by setting `h.visible` to `0`. It checks the visibility setting in line 3 by examining `h.visible`.

```
h = actxserver('Matlab.Application');
h.visible = 0;

h.visible
ans =
     0
```

**MATLAB Engine.** For a MATLAB engine session, use the `engSetVisible` and `engGetVisible` functions that are new in MATLAB 6.1. Line 4, below, disables the visibility of the MATLAB engine window using `engSetVisible` with an argument of `0`. Line 5 checks this setting with `engGetVisible`.

```
Engine *ep;
bool vis;
ep = engOpen(NULL);
engSetVisible(ep, 0);
engGetVisible(ep, &vis);
```

## Serial I/O

MATLAB's serial port interface provides direct access to peripheral devices such as modems, printers, and scientific instruments that you connect to your computer's serial port. This interface is established through a serial port object, which you create with the `serial` function.

**Freeing the Serial Port on Windows Platforms.** The serial port object uses the `javax.comm` package to communicate with the serial port. However, due to a memory leak in `javax.comm`, the serial port object is not released from memory. You can use the `freeserial` function to release the MATLAB hold on the serial port.

`freeserial` is necessary only on Windows platforms. You should use `freeserial` only if you need to connect to the serial port from another application after a serial port object has been connected to that port, and you do not want to exit MATLAB.

**Events, Callbacks, and Function Handles.** Action properties and action functions are now referred to as callback properties and callback functions. This new terminology is reflected in new names for the associated properties and functions. The general rule for the name changes is to change "Action" to "Fcn" for properties, and "action" to "callback" for functions. For example, `TimerAction` has been renamed `TimerFcn`, and `instraction` has been renamed `instrcallback`.

Additionally, if you want to automatically pass the object and event information to the callback function, then you must specify the function as either a function handle or as a cell array. Note that you can also specify the callback function as a string. In this case, the callback is evaluated in the MATLAB workspace and no requirements are made on the function's input arguments.

**Enhancements to Existing Properties.**

- `Terminator` Property – You can configure `Terminator` to a decimal value ranging from 0 to 127, to the equivalent ASCII character, to `CR/LF` or `LF/CR`, or to empty (').
- Timer events – Some timer events may not be processed if your system is significantly slowed or if the `TimerPeriod` value is too small. The minimum `TimerPeriod` value is now 0.01 second.

## Creating Graphical User Interfaces — GUIDE

This section lists the changes made to GUIDE for Release 12.1:

- The Layout Editor **Edit** menu has **Undo** and **Redo** items. You can undo or redo layout actions and property settings (with the exception of the figure `FileName` property).
- The **Application Option** dialog supports a new option for **Command-line accessibility** – Callback. This option is now the default.
- The Layout Editor displays the layout grid in the current figure color.
- The Layout Editor context menus have been reorganized.
- The Menu Editor enables you to rearrange the order of menu items.
- The Menu Editor adds callback function stubs to the application M-file.

See Creating Graphical User Interfaces in the MATLAB documentation for more information.

# Major Bug Fixes

MATLAB 6.1 includes several bug fixes made since MATLAB 6.0. This section describes the particularly important bug fixes.

Also, MATLAB 6.1 includes several important MATLAB 6.0 bug fixes.

## Development Environment

### Help Browser Supports Mouse Wheel

For Windows platforms, the wheel on your mouse will now work in the Help browser.

### UNIX Help Browser Search Results Now Highlighted

On UNIX systems, when you perform a full text search using the Help browser, the search terms are highlighted when you view a page.

### UNIX Paste Problems Fixed

On some UNIX systems, pasting after a cut or copy would sometimes cause the system to hang. That problem has been fixed. However, due to issues with UNIX itself, the paste does not always work and you might have to do it again.

## Mathematics

### Memory Leak Fixed in Matrix Multiply

Under certain conditions, matrix multiply (which includes matrix-vector multiply, vector-matrix multiply, and even vector inner products) leaked memory. For example, on a Pentium III under Linux or Windows, any vector inner product of length greater than 15,000 leaked memory. This was observed by MATLAB increasing its use of system resources that were never returned. MATLAB 6.1 uses new ATLAS BLAS libraries that no longer leak memory.

### Improved Convergence for eigs(A,k,'sm') and eigs(A,k,0)

In MATLAB 6.0, `eigs` was reimplemented to use the ARPACK library of routines. Unfortunately, the smallest magnitude case, `sigma = 'sm'` and `sigma = 0`, chose the wrong algorithm. For MATLAB 6.1, the correct ARPACK algorithm is used and convergence is much quicker.

This bug fix introduces a backwards incompatibility. When A is a function `Afun` and `sigma = 'sm'`, `Afun` must now return `Y = A\x`. Prior to MATLAB 6.1, `eigs` required `Afun` to return `y = A*x` for this case.

### quad Sampling Improved

In MATLAB 6.0, `quad('cos(4*n*x)',-pi,pi)` returned `2*pi` instead of `0`. When `quad` initially sampled the function, it incorrectly assumed the function is the constant 1 over the interval `[-pi,pi]` and so returned `2*pi` early. It now samples more carefully and returns `0`.

### griddata3 Inner Matrix Error Message

In MATLAB 6.0, an internal error sometimes caused `griddata3` to display the error message, `Inner matrix dimensions must agree`. This error has been corrected.

### Improved Handling of Degenerate Triangulation

In MATLAB 6.0, there were sometimes problems associated with degenerate triangulation. For example, `convhull` could produce a convex hull that did not cover all the original data. MATLAB 6.1 corrects this problem by replacing the utility function `delaunayc` with Qhull.

### Error Message Display for Qhull-Related Functions

In MATLAB 6.0, Qhull-related functions (e.g., `delaunayn`) displayed error messages in standard error. For UNIX platforms, standard error is different from the command window. For MATLAB 6.1, error messages are displayed in the command window.

### histc Computes First Two Bins Correctly

Prior to MATLAB 6.0, `histc` produced the wrong results for the first two bins for cases with extremely nonuniform bin edges. This problem was corrected in MATLAB 6.0.

# Platform Limitations

**Note** The Release 12.0 platform limitations for MATLAB for the HP and IBM platforms still apply to Release 12.1, with the exception of the specific differences highlighted below.

The MATLAB functionality described in these *Release Notes* and in the MATLAB documentation applies to MATLAB 6.0 with the exception of the limitations listed below for the HP and IBM platform. For background information, see "Platform Limitations" on page 1-17:


- "Patch Required for HP-UX 11.0" on page 2-20
- "Mathematics Limitations" on page 2-20
- "Graphics and Printing Limitations" on page 2-21
- "GUIDE Limitations" on page 2-21


Another platform limitation involves the use BLAS on certain processors. See "You May Need to Overwrite the MATLAB Default Choice of BLAS" on page 2-22 for details.

## Patch Required for HP-UX 11.0

To run MATLAB on HP-UX 11.0, you must install a patch available from Hewlett-Packard. To get the patch, go to `www.itrc.hp.com`, the IT Resource Center page. The patch is available to registered customers from the individual patches link. The patch name is below.
PHSS_21959 1.0 X/Motif 32 bit
Runtime 2000 Periodic Patch.

## Mathematics Limitations

The Basic Fitting interface is not supported. Instead, use curve fitting functions such as `polyfit` and `spline`. See also "Data Analysis and Statistics" in the MATLAB documentation for more information.

## Graphics and Printing Limitations

| Feature | Limitation and Alternatives |
|---|---|
| Data Statistics | Not supported. |
| Printing | Uses the Release 11 **Page Setup**, **Print Setup**, and **Print** dialog boxes. For information about these interfaces, see "Printing MATLAB Graphics" in the online MATLAB documentation. |
| Property Editor | Not supported. Similar graphical user interfaces provide access to figure, line and text objects. Use the `set` and `get` functions to modify Handle Graphics object properties. |

## GUIDE Limitations

The **Application Options** dialog configures the M-file generated by GUIDE. If you comment out lines of code in this M-file, the **Application Options** dialog may not be able to represent the state of the generated code correctly. In addition, the M-file may not function correctly.

If you want to remove code that has been added due to the settings of the **Application Options** dialog, you should make changes to the dialog. GUIDE updates the M-file the next time you save your layout.

If you want to prevent certain lines of generated code from executing, you should remove these lines from the M-file instead of making these lines into comments.

Note that these limitations apply only to code generated by GUIDE, not user-written code.

## You May Need to Overwrite the MATLAB Default Choice of BLAS

On the PC, under both Linux and Windows operating systems, MATLAB determines at startup time what processor your computer has, for example Genuine Intel Pentium II, Pentium III, or AMD Athlon. MATLAB then automatically selects the most appropriate BLAS for your processor. The same is true on the SUN, where MATLAB distinguishes between UltraSPARCs and non-Ultra machines.

However, on the remaining platforms you get the default BLAS, which is usually targeted for a reasonably modern or common processor. Namely:

- ALPHA 21264
- HP700 PA-RISC1.1
- HPUX PA-RISC2.0
- IBM_RS Power3
- SGI R12000

If you have reason to believe that your processor is closer to another of the flavors of BLAS distributed with MATLAB, for example 21164 on the ALPHA or PA-RISC2.0 on the HP700, you might want to override the default choice of BLAS. Look in your `<MATLAB>/bin/$ARCH` directory for libraries beginning with `atlas_` to see your options.

### Overriding the Default

The way to override the default choice is to set the environment variable `BLAS_VERSION` before invoking MATLAB. For example (in `csh`)

```
setenv BLAS_VERSION  atlas_21164.so
setenv LAPACK_VERBOSITY 1
matlab
```

The environment variable `LAPACK_VERBOSITY` simply confirms that your choice of BLAS is being loaded once you start up MATLAB.

### Restoring the Default

If you would like to return to using the default provided by MATLAB, you may use the command (in `csh`)

```
unsetenv BLAS_VERSION
```

### Determining Your Processor

This section indicates how you can determine what type of processor your computer has.

**ALPHA.** Enter the command

```
/usr/sbin/psrinfo -v
```

which returns something like

```
Status of processor 0 as of:05/03/01 17:16:55
  Processor has been on-line since 04/20/2001 14:09:31
  The alpha EV4.5 (21064) processor operates at 233 MHz,
  and has an alpha internal floating point processor.
```

The number in parentheses on the third line, in above example (21064), is the number you are interested in.

**GLNX86.** Enter the following command

```
cat /proc/cpuinfo
```

and look for the following fields in the output (values may vary from the example below)

```
vendor_id  :GenuineIntel
cpu family :6
model      :8
model name :Pentium III (Coppermine)
stepping   :1
```

Match up this information with the table in `<MATLAB>\bin\glnx86\blas.spec`.

---

**Note** Some versions of glibc 2.1.x have problems with environment variables (and the ability to reliably query them) from within shared library `init` functions. To take advantage of the `BLAS_VERSION` feature, you may need to upgrade your machine to glibc 2.2.

---

**HP700.** Start with the **System Administration Manager (SAM)** and work your way to the **Processor** tab, as shown below:

**System Administrator Manager (SAM)** -> **Performance monitors** -> **System properties** -> **Processor** tab

This provides information about the type of processor.

**HPUX.** MATLAB only supports HPUX running on PA-RISC2.0.

**IBM_RS.** Contact IBM Technical Support and request the document entitled "Determining CPU Speed in AIX." This is a table of machine types, processor types, and processor speeds.

**SGI.** Enter the following command

```
sysinfo -a
```

which returns a lot of information. In the first few lines, look for information something like

```
CPU Type is              mips R4400 5.0
```

The information starting with `R` is what you are interested in. MATLAB ships for the `R5000`, `R8000`, `R10000` and `R12000` (default).

**SOL2.** Enter the following command

```
uname -m
```

which returns either `sun4u` for UltraSPARC or `sun4m` for the older, non-Ultra machines (e.g., Hyper and SuperSPARCs).

**WIN32.** Start with the **My Computer** icon, and work your way to the **General** tab, as shown below:

**My Computer** -> **Control Panel** -> **System** -> **General** tab

This should list the family and model number for your computer. On Windows NT and Windows 2000, the same information is on the **Environment** tab, under the System Variable `PROCESSOR_IDENTIFIER`. Match up this information with the table in `<MATLAB>\bin\win32\blas.spec`.

### Using Another BLAS

You may also use BLAS from other sources than the ones shipped with MATLAB, provided they are in the correct format. This format is a shared library (as opposed to a static library) that exports all the double-precision (starting with d) and double-precision complex (starting with z) BLAS routines from `dasum` to `zupmtr`. On HP, IBM_RS, and WIN32, the symbols must be exported without trailing underscores, while for ALPHA, GLNX86, SGI, and SOL2, the symbols must be exported with trailing underscores (e.g., `dgemm_` ).

If the shared library you provide also includes LAPACK symbols like `dgefa` (or `dgefa_` ), then they will override the MATLAB default implementation, which is based on the Fortran LAPACK from Netlib at `http://netlib.org`.

# Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from MATLAB 6.0 (Release 12.0) to MATLAB 6.1 (Release 12.1).

For information about upgrading from an earlier version than MATLAB 6.0, see the *Release Notes for Release 12*.

This section about upgrading from an earlier release is organized into the following subsections:


- "Development Environment Issues" on page 2-26
- "Mathematics Issues" on page 2-27
- "Programming and Data Types Issues" on page 2-28
- "Graphics Issue" on page 2-29
- "External Interfaces/API Issues" on page 2-30


## Development Environment Issues

### subscribe Function No Longer Supported

The `subscribe` function is no longer supported.

### Command History, Preferences, and Favorites

If you uninstall Release 12.0, you will lose the Command History, preferences, and Help browser favorites from Release 12.0. For instructions to maintain these files when uninstalling Release 12.0, see "Maintaining Preference Files After Uninstalling" in Chapter 1 of the *Release Notes*.

### Help Browser Favorites

If you use favorites you created for the documentation in the Release 12.0 Help browser, those favorites may point to an incorrect or invalid location in Release 12.1. You will need to delete any invalid favorites and add those favorites again.

### Source Control

If you use Microsoft Visual SourceSafe with MATLAB's source control features, you now need to specify the login information for SourceSafe using preferences. Select **File -> Preferences -> General -> Source Control** from the desktop. Specify the **Username**, **Password**, and **Database**.

## Mathematics Issues

### Finding Smallest Magnitude Eigenvalues

`eigs(A,k,sigma)` and `eigs(A,B,k,sigma)` return `k` eigenvalues based on `sigma`. For `sigma = 'sm'`, `eigs` returns the smallest magnitude eigenvalues.

In MATLAB 6.0, `eigs` was reimplemented to use the ARPACK library of routines. Unfortunately, the smallest magnitude case, `sigma = 'sm'` and `sigma = 0`, chose the wrong algorithm. For MATLAB 6.1, the correct ARPACK algorithm is used and convergence is much quicker.

This bug fix introduces a backwards incompatibility. When `A` is a function `Afun` and `sigma = 'sm'`, `Afun` must now return `Y = A\x`. Prior to MATLAB 6.1, `eigs` required `Afun` to return `y = A*x` for this case.

### Possible Changes in Results Returned by Matrix Functions

Starting in MATLAB 6.0 (R12.0), matrix computations are based on LAPACK, a large, multiauthor Fortran subroutine library for numerical linear algebra. While this change has many benefits (see the *Release Notes for Release 12*) and matrix functions continue to operate in the same way in MATLAB 6.0, the results returned by matrix functions may differ. Changes in roundoff errors can be seen in most matrix computations. In cases where quantities are not uniquely determined mathematically, results may differ in order and in normalization.

For example:

- Eigenvalues may be returned in a different order.
- Eigenvectors may be normalized differently.
- The signs of columns of orthogonal matrices may differ.
- `rcond` is a better estimate of the reciprocal condition.
- `lu` can now be used to factor rectangular full matrices.

### Obsolete Input Arguments

Certain input arguments to these functions have become obsolete. Using these arguments does not result in an error, but they are ignored.

| Function | Description |
|---|---|
| `delaunay` | Now ignores the third argument `fuzz`, which specified a value for the `fuzz` standard deviation.<br><br>Now ignores the third argument `'sorted'`. This argument indicated to `delaunay` that the given points `x` and `y` were sorted, and that duplicate points had been eliminated. |
| `convhull` | Now ignores the third argument `TRI`, which provided triangulation data previously computed using `delaunay`. |

### Obsolete Functions

The following MATLAB function has become obsolete. For backwards compatibility, it has not been removed from the language at this time. However, this function may be removed in a future release, and you are encouraged to discontinue its use, or use the function that replaces it.

| Function | Description |
|---|---|
| `bvpval` | Evaluate the numerical solution of a boundary value problem (BVP). Replace with `deval`, which evaluates the solution of both initial value and boundary value differential equation problems. |

## Programming and Data Types Issues

### Output from Background and Foreground Commands (UNIX)

In Release 12 on UNIX platforms, a background command (i.e., any system command after which you add a `&`), such as

```
! cat startup.m &
```

no longer produces any output. Prior to Release 12, a background command sent output to the command window.

If you need to see the output from a command, either do not make the command a background command (i.e., remove the &), or run the background command in a separate xterm. To start another xterm, issue the following command.

```
! xterm &
```

In Release 12, foreground functions (i.e., nonbackground functions) send their output to the diary, if the `diary` function has been issued. The output is also displayed in the command window (prior to Release 12, foreground function output was only displayed in the command window).

### matlab_helper Process

To make the `!` and `unix` commands operate more efficiently, in Release 12 MATLAB creates a secondary process, called `matlab_helper`, at startup.

This `matlab_helper` contains those elements of MATLAB necessary to run the `!` and `unix` commands.

## Graphics Issue

### MATLAB No Longer Supports Terminal Mode

MATLAB no longer runs on nongraphics computer terminals.

# External Interfaces/API Issues

## Concatenation of Java Objects

When you concatenate Java objects, the class of the resultant object depends on the classes of the input objects, as follows:

- If the input objects are of the same class, MATLAB makes the output object of that class. This was true in Version 6.0 as well.
- If the input objects are of different classes, but all inherit from a common class, MATLAB makes the output object of the common parent class. MATLAB selects the lowest common parent in the Java class hierarchy as the output class. This is new behavior for Version 6.1.

  For example, concatenating objects of classes `java.lang.Integer` and `java.lang.Double` creates a new object of class `java.lang.Number`.
- If the input objects are of different and unrelated classes, then MATLAB makes the output object of the `java.lang.Object` class. This was true in Version 6.0 as well.

## Obsolete Fortran MX, MEX, MAT, and ENG Functions

The following Fortran MX, MEX, MAT, and ENG functions are considered to be obsolete as of Version 6.1. Support for these functions may be removed from a future MATLAB release.

**Table 2-5: Obsolete Fortran MX Functions**

| | |
|---|---|
| `mxCreateFull` | `mxFreeMatrix` |
| `mxIsFull` | `mxIsString` |

**Table 2-6: Obsolete Fortran MEX Functions**

| | |
|---|---|
| `mexGetEps` | `mexGetGlobal` |
| `mexGetFull` | `mexGetInf` |
| `mexGetMatrix` | `mexGetMatrixPtr` |
| `mexGetNaN` | `mexIsFinite` |

**Table 2-6: Obsolete Fortran MEX Functions (Continued)**

| | |
|---|---|
| mexIsInf | mexIsNaN |
| mexPutFull | mexPutMatrix |

**Table 2-7: Obsolete Fortran MAT Functions**

| | |
|---|---|
| matDeleteMatrix | matGetFull |
| matGetMatrix | matGetNextMatrix |
| matGetString | matPutFull |
| matPutMatrix | matPutString |

**Table 2-8: Obsolete Fortran Engine Functions**

| | |
|---|---|
| engGetFull | engGetMatrix |
| engPutFull | engPutMatrix |

# Known Software and Documentation Problems

This section updates the MATLAB 6.1 documentation set, reflecting known MATLAB 6.1 software and documentation problems. It is organized into the following subsections:


- "Development Environment Problems" on page 2-32
- "Documentation Updates" on page 2-33


## Development Environment Problems

### Displaying Results From lookfor Function

When you run the `lookfor` function, press **Ctrl+C** to display the results in the command window.

### Many Open Windows Can Cause a Crash or Hang (Windows 95/98/Me)

On Microsoft Windows 95/98/Me platforms, if you keep many windows open, MATLAB may crash or hang. For example, if you keep open 12 Stateflow windows, or 25 figure windows, or 50 Simulink windows, you may experience this problem. Note that these numbers are only estimates; the actual number of open windows that may cause this problem depends on the resources currently in use by other components and applications.

### Cannot Go To Top Level of UNC Path

For Windows platforms, you cannot use `cd` or any directory tool in the MATLAB desktop (including the Current Directory browser and Set Path dialog box) to access the top level of a UNC path.

### Workspace Browser with Many Variables

If there are many variables in the workspace and the Workspace browser is open, you may experience performance problems. If you expect to have more than 1000 variables in the workspace, close the Workspace browser to avoid performance problems.

### UNIX Display Problems When UNIX Client and Server Platforms Differ

If you use MATLAB on UNIX and the platform for the server is different than that for the client, there may be problems with the display of graphics on the

client. See the Technical Support Web page for a solution that lists the combinations tested and any known display problems with them.

### Sun Solaris 16-Bit Display Not Supported

Sun's Java VM for Solaris does not support 16-bit displays. Therefore you cannot use this configuration with Release 12. Use another display mode instead.

### Sun Solaris Arrow Keys Not Working

On some Sun Solaris systems, the arrow keys on the main keyboard are not working properly. Instead, try the arrow keys in the numeric keypad.

### Alpha Shortcut Problems When Using Emacs Key Bindings in Editor

On the Alpha platform, if you set the Editor/Debugger preference for key bindings to Emacs, the shortcuts for **Undo** (**Ctrl+_**) and **Copy** (**~+W**) do not work.

### Display Problems with Xoftware

If you use Xoftware on a PC to run MATLAB on a UNIX platform, you need to do the following to avoid display problems:

1. Go to the Xoftware **Control Panel**.
2. From the **Options** menu, select **Configuration**.
3. Select the **Window** tab.
4. From the **Options** listing, select **Concurrent Window Manager**.
5. Under **Settings**, select **Off**.
6. Click **OK**.

## Documentation Updates

### Editor/Debugger Example - Graphic and Information Incorrect

In the printed book *Using MATLAB* (Version 6), on page 7-19, the graph shown is incorrect. For the correct graph, see the same page in the Help browser at

**MATLAB -> Using MATLAB -> Development Environment -> Editing and Debugging M-Files -> Debugging M-Files -> Trial Run for Example**.

On page 7-29, in "Correcting Problems and Ending Debugging, Completing the Example," step 3 is incorrect. It should instead read "In `collatzplot.m` line 12, change the string `plot_seq` to `seq_length(m)` and save the file."

### interp1 Extrapolation of Out-of-Range Values

A new argument enables `interp1` to perform extrapolation for out-of-range values for all methods. It also enables you to specify a scalar to be returned for out-of-range values.

The PDF version of the `interp1` reference page incorrectly states that the default for all methods is for `interp1` to perform extrapolation for out-of-range values. In fact, `interp1` performs extrapolation as the default only for the `'spline'`, `'pchip'`, and `'cubic'` methods. For all other methods, it returns `NaN` for out-of-range values. This behavior is unchanged from Version 5.

The HTML reference page for `interp1` is correct.

3

# Simulink 4.1 Release Notes


**New Features** . . . . . 3-2
Simulink Editor . . . . . 3-2
Modeling Enhancements . . . . . 3-4
Simulink Debugger . . . . . 3-7
Block Library . . . . . 3-8

**Bug Fixes** . . . . . 3-10

**Platform Limitations for HP and IBM** . . . . . 3-12

**Upgrading from an Earlier Release** . . . . . 3-13
Running Simulink 4.1 Models in Simulink 4.0 . . . . . 3-13
Simulink Block Library Reorganization . . . . . 3-13
Direct Feedthrough Compensation Deprecated . . . . . 3-13
S-Functions Sorted Like Built-In Blocks . . . . . 3-14
Added Latched Triggered Subsystems . . . . . 3-14
Self-Triggering Subsystems Are No Longer Allowed . . . . 3-14
Improved Invalid Model Configuration Diagnostics . . . . . 3-15

**Known Software Problems** . . . . . 3-16
Accelerator Mode Does Not Support Inline Parameters . . . 3-16
Turn the New Wrap Lines Option Off . . . . . 3-16

# New Features

This section introduces the new features and enhancements added in Simulink 4.1 since Simulink 4.0 (Release 12.0).

For information about Simulink features that are incorporated from recent releases, see the *Release Notes for Release 12*.

---

**Note** For information about closely related products that extend Simulink, see the sections on the Simulink Performance Tools and the Real-Time Workshop, as well as information about the blocksets from The MathWorks.

---

This section about new Simulink features is organized into the following subsections:


- "Simulink Editor" on page 3-2
- "Modeling Enhancements" on page 3-4
- "Simulink Debugger" on page 3-7
- "Block Library" on page 3-8


## Simulink Editor

This section describes enhancements to the Simulink Editor.

### Undo Move

In Simulink 4.1, the **Undo** command on the Simulink **Edit** menu restores blocks, annotations, lines, and nodes that have moved to their original locations (see "Undoing a Command" in *Using Simulink*).

### Undo Subsystem Creation

In Simulink 4.1, the **Undo** command on the Simulink **Edit** menu restores blocks that have been grouped into a subsystem to their original level in the model (see "Undoing Subsystem Creation" in *Using Simulink*).

### Autoconnecting Blocks

This version makes connecting blocks significantly easier. To connect a set of source blocks to a target block, simply select the source blocks, hold down the **Ctrl** key and left--click the target block. Simulink draws connecting lines between the source blocks and the destination block, neatly routing lines around intervening blocks. To connect a source block to a set of target blocks, select the target blocks, hold down the **Ctrl** key and left--click the source block. To connect two blocks, select the source block, and left-click the destination block while holding down the **Ctrl** key. Simulink connects as many ports on the two blocks as possible (see "Autoconnecting Blocks" in *Using Simulink*).

### Autorouting Signal Lines

Simulink now routes signal lines around intervening blocks when you connect them either interactively (by dragging the connecting lines or using autoconnect) or programmatically via the `add_line` command's new `'autorouting'` option (see "Autorouting Option Added to add_line Command" on page 3-4).

### Displaying Storage Class on Lines

This version adds an item to the **Format** menu, which toggles the display of (nonAuto) storage class on signal lines (see "RTW Storage Class" in *Using Simulink* for more information).

### Save Models in Release 11 Format

This release can save post-Release 11 models in Release 11 format. Simulink 3 (Release 11) can load and run converted models that do not use any post-Release 11 features of Simulink. Simulink 3 can load converted models that use post-Release 11 features but may not be able to simulate the model correctly. Use the **Save as** option from the Simulink **File** menu or the following command to save a model in Release 11 format.

```
slsaveas(SYS)
```

See "Saving a Model in Simulink 3 (R11) format" in *Using Simulink* for more information.

## Modeling Enhancements

This section describes enhancements to Simulink's dynamic system modeling tools.

### Autorouting Option Added to add_line Command

The `add_line` command now optionally routes lines around intervening blocks and annotations. For example, the following command autoroutes a connection between two blocks in the `vdp` model.

```
add_line('vdp','Product/1','Mu/1','autorouting','on')
```

The autorouting option is off by default. See `add_line` in *Using Simulink* for more information.

### S-Function Builder

The S-Function Builder generates an S-function from specifications that you enter in a dialog box. It provides an easy way for you to incorporate existing code into a Simulink model.

### add_param, delete_param

With this version, you can add custom parameters to your block diagrams.

```
add_param('modelname','MyParameterName','value')
delete_param('modelname','MyParameterName')
```

You can also use the model handle in place of the model name. See `add_param` and `delete_param` in *Using Simulink* for more information.

### Connection Callbacks

With this version, you can use `set_param` to set callbacks on ports that are triggered by changes in the ports' connectivity. The callback function parameter is named `ConnectionCallback`. When the port's connectivity changes (addition/deletion of line connected to the port, connection of new block to the port, etc.), Simulink invokes the callback function with the port handle as its argument. See "Port Callback Parameters" in *Using Simulink* for more information.

### Saving Block User Data in Model Files

This version adds a new block parameter, named `UserDataPersistent`, that is `off` by default. Setting this parameter `on`, e.g.,

```
set_param(block-name,'UserDataPersistent','on')
```

causes Simulink to include a block's user data (i.e., the value of the block's `UserData` parameter) in the model file when you save a model. Simulink encodes the user data as ASCII characters and saves the encoded data in a new section of the model file called `MatData`. This mechanism works with all forms of MATLAB data, including arrays, structures, objects, and Simulink data objects. See "Associating User Data with Blocks" in *Using Simulink* for more information.

### Absolute Tolerance Enhancements

This version adds a dialog item for setting the absolute tolerance for each state in the State-Space block, the Transfer Fcn block, and the Zero-Pole block. With this enhancement, you can now specify the absolute tolerance for solving every continuous state in your model.

### Block Reduction Enhancements

S-functions may now request that they be eliminated from the compiled model. To do this, call `ssSetBlockReduction(true)` inside the S-function. This is an advanced feature provided for customers writing S-functions who want to optimize the generated code produced for their S-function. Graphical connectivity is now remapped during block reduction, eliminating a source of error during reduction (e.g., a memory reference error used to occur if Simulink eliminated a block connected to a scope). Block reduction is now `on` by default, and a Simulink preference has been added for the option.

### Boolean Logic Signals Preference

The Simulink Preferences dialog box now allows you to specify the use of Boolean logic signals by default. See "Setting Simulink Preferences" in *Using Simulink* for more information.

### Subsystem Semantics Demos

Typing `sl_subsys_semantics` at the MATLAB prompt now displays a set of models that illustrate the semantics of various types of subsystem blocks. The demos include formal definitions of function-call subsystems.

### Enhanced Engine Model Demos

The top and bottom dead center detection in the `engine` and `enginewc` demo models now use a reset integrator. In previous versions, the models used a triggered subsystem to detect angular position. This method resulted in inefficiencies and a slower, less accurate solution. In addition, self-triggering subsystems are now illegal in Simulink.

### Setting Block Sorting Priority on Virtual Subsystems

In Simulink 4.0, it was an error to specify a priority on a virtual subsystem. In Simulink 4.1, you can specify priorities on virtual subsystems.

### Using ~ in Filenames on UNIX

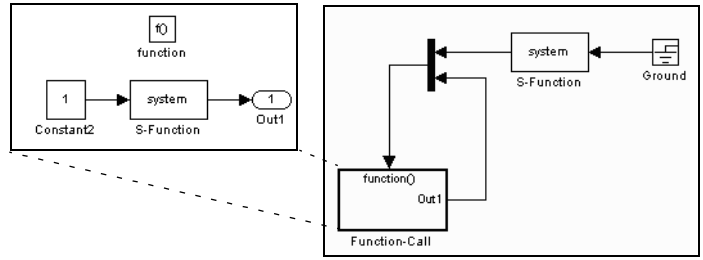
Now all filename fields in Simulink support the mapping of the ~ character in filenames. For example, in a To File block, you can specify `~/outdir/file.mat`. On most systems, this will expand to `/home/$USER/outdir/file.mat`.

### Improved Warning About Slow Signals Feeding the Enable Port of an Enabled Subsystem Containing Fast Blocks

In a multitasking environment, deterministic results cannot be guaranteed if a slow signal feeds the enable port of an enabled subsystem that contains fast blocks. In previous versions, Simulink did not issue a warning in some cases where this may occur.

### Flagging Function-Call Subsystem Cycles

In previous versions, Simulink allowed you to build models containing function-call-cycles, i.e., function-call subsystems that directly or indirectly call themselves.



Such models cannot be correctly simulated. Accordingly, Simulink now displays an error message when you attempt to run or update a diagram containing function-call cycles.

## Simulink Debugger

This section describes enhancements to the Simulink debugger.

### Enhancement to Sorted List Display

The Simulink debugger (`sldebug`) sorted list command, `slist`, now displays the names of the S-functions residing inside S-function blocks.

### Improved Messages in Accelerated Mode

The `trace`, `break`, `zcbreak`, `nanbreak`, and `minor` commands now indicate that they are disabled when in accelerator mode and you need to switch to normal mode to activate them. The spacing of several messages has been fixed so the text aligns correctly.

### Breakpoints on a Function-Call Subsystem

You can now put a break point on a function-call subsystem. Simulink breaks when the subsystem is executed. In Release 12, entering the `quit` command while at a breakpoint within a function-call subsystem wouldn't always quit the debugger. Now the `quit` command ends the debugging session once the initiating (calling) Stateflow chart or S-function finishes executing its time step.

### Displaying and Probing Virtual Blocks

The `display` and `probe` commands now work for virtual blocks.

### Stepping Stateflow Charts

You can now step execution of a model into a Stateflow chart.

## Block Library

This section describes enhancements to the Simulink block libraries.

### Unified Pulse Generator

This version merges the Discrete Pulse Generator block into the Pulse Generator block. The combined block has two modes: time-based and sample-based (discrete). Time-based mode varies the step size when a variable step solver is being used to ensure that simulation steps occur at pulse on/off transitions. When a fixed step solver is used, the time-based mode computes a fixed step size that ensures that a simulation step occurs at every pulse transition. The Pulse Generator block also outputs a pulse of any real data type in sample-based as well as time-based mode.

### Control Flow Blocks

Simulink 4.1 adds an If block and Switch Case block that can drive conditionally executed subsystems that contain instances of the new Action Port block. Action subsystems are similar to enabled subsystems, except that all blocks must run at the same rate as the If or Switch Case block.

This version also adds a For Iterator block and a While Iterator block. When placed in a subsystem, these blocks cause all of the blocks in the system to run multiple cycles during a time step. The block cycle in a For Iterator subsystem runs a specified number of times. The block cycle in a While Iterator subsystem runs until a specified condition is false. A user can limit execution of a While Iterator subsystem to a specified number of iterations to avoid infinite loops.

The new Assignment block allows a model to assign values to specified elements of a signal.

### Bus Creator

Simulink 4.1 adds a Bus Creator block that combines the output of multiple blocks into a single signal bus. A model can use the existing Signal Selector block to extract signals from the bus. The block's dialog box allows you to assign names to signals on the bus or allow the signals to inherit their names from their sources. When you double-click on a signal name in the block dialog, the source block is highlighted. There is no execution overhead in the use of bus creator/bus selector blocks.

### Sine Wave Block Enhancements

The Sine Wave block now supports a bias factor that eliminates the need to sum with a Constant block. The Sine Wave block also has a new computational mode. This mode (called sample-based) eliminates the dependence on absolute time.

### Enhanced Flip-Flop Blocks

Simulink Extras (`simulink_extras.mdl`) contains a set flip-flop blocks. These blocks now use the new triggered subsystem latching semantics. In addition, the S-R Flip-Flop block now models a physical `NOR` gate (i.e., `S=1`, `R=1 => Q=0`, `Q!=0`, the undefined state).

### Additional Data Type Support

The Discrete-Time Integrator and Rounding Function blocks now handle `single` as well as `double` values. The Transport Delay, Unit Delay, Variable Transport Delay, Memory, Merge, and Outport blocks can specify nonzero initial conditions when operating on fixed-point signals.

### Simulink Block Library Reorganization

The Simulink Block Library contains a new Subsystems sublibrary. The new library contains most of the new control flow blocks as well as subsystem and subsystem-related blocks that used to reside in the Signals & Systems library. The subsystems in the new library each contain the minimum set of blocks needed to create a functioning subsystem, e.g., an input port and an output port.

### Scope Enhancements

The Scope block includes the following enhancements:

- A floating version of the Scope added to the Sinks block library
- Floating Scope saves the signals selected for display in the model file
- The Scope's toolbar buttons for toggling between floating/nonfloating mode, restoring saved axes, locking/unlocking axes, and displaying the **Signal Selector**

# Bug Fixes

This section lists fixes to bugs that occurred in the previous version of Simulink.

- Variable sample time S-functions

  Simulink no longer crashes when an S-function with variable sample time is placed in an atomic subsystem.

- Bus selector detection of duplicated names

  A bug related to the detection of a duplicated name in a bus that was feeding a Bus Selector block was fixed.

- Optimize block memory use

  In Simulink 4.0, the Continuous and Discrete Transfer Function blocks and the Discrete Filter block used more memory than they needed to, particularly for the case of many poles. They now use an optimal amount of memory.

- Miscellaneous fixes to the model loader

  Miscellaneous bug fixes have been performed on the model loader:

  - The loader and saver now retain any comment lines (i.e., lines that begin with #) that are found at the top of the model file.
  - The loader does not crash on Windows NT when file sizes are integer multiples of 4096.
  - The loader does not hang on corrupt models in which blocks with duplicate names are found.

- Profiler fixes

  The Simulink profiler now saves its files in the temporary directory. See the MATLAB command `tempdir`. The help was also updated.

- Chirp block fix

  The Chirp block now sweeps through frequencies correctly from the initial frequency at the simulation start time to the target frequency at the target time.

- Function-call subsystem bug fixes

  This version fixes several bugs related to the execution orders of function-call subsystems.

- Sorting bug fix

  Previous versions incorrectly computed the direct feedthrough setting for nonvirtual subsystems in triggered/function-call subsystems. This resulted in incorrect execution (sorting) orders. Now all nonvirtual subsystems within triggered subsystems have their direct feedthrough (needs input) flags set for all input ports. This is needed because a nonvirtual subsystem with a triggered sample time executes both its output and update methods together within the context of the model's output method.

- Fixed handling of grounded/unconnected inputs feeding certain blocks

  Simulink 4.0 incorrectly handled grounded or unconnected inputs to level-1 and level-2 S-functions requiring contiguous inputs and to some Matrix blocks. This has been fixed in Simulink 4.1.

# Platform Limitations for HP and IBM

**Note** The Release 12 platform limitations for Simulink for the HP and IBM platforms still apply to Release 12.1. In addition, there are more limitations, described below.

The following Java-dependent Simulink features, introduced in Simulink 4.1, are *not* available on these platforms.

- Simulink Data Class Designer
- S-Function Builder
- Look-Up Table Editor

# Upgrading from an Earlier Release

This section discusses upgrade issues in moving from Simulink 4.0 to Simulink 4.1.

See the *Release Notes for Release 12* for upgrade issues involved in moving from Simulink 3.0 (Release 11.0) to Simulink 4.1.

## Running Simulink 4.1 Models in Simulink 4.0

Simulink 4.0 can run models created or saved by Simulink 4.1 as long as the models do not use features introduced in the new version, including new block types and block parameters. In particular, you should not attempt to use Simulink 4.0 to simulate or even open models that use the new Simulink control flow blocks. Opening such models cause Simulink 4.0 to crash.

## Simulink Block Library Reorganization

The Simulink Block Library contains a new Subsystems sublibrary. The new library contains most of the new control flow blocks as well as subsystem and subsystem-related blocks that used to reside in the Signals & Systems library. The subsystems in the new library each contain the minimum set of blocks needed to create a functioning subsystem, e.g., an input port and an output port.

## Direct Feedthrough Compensation Deprecated

If an S-function needs the current value of its input to compute its output, it must set its direct feedthrough flag to true. Previously, if a direct feedthrough S-function failed to do this, Simulink tried to provide a valid signal to the S-function's `mdlOutput` (M-file `flag=3`) or `mdlGetTimeOfNextVarHit` (M-file `flag=4`) methods. This special compensation mode for S-functions was flawed. For this reason, the current version deprecates the mode, though making it available as an option. In this version, by default, if an S-function sets its direct feedthrough flag to false during initialization, Simulink sets the S-function's input signal to `NULL` (or a `NaN` signal for M-file S-functions) during the `mdlOutput` or `mdlGetTimeOfNextVarHit` methods. Thus, in this version, models with S-function(s) may produce segmentation violations. See *`matlabroot`*`/simulink/src/sfuntmpl_directfeed.txt` for more information.

## S-Functions Sorted Like Built-In Blocks

When sorting blocks, Simulink now treats S-function blocks the way it treats built-in blocks. This means that S-functions now work correctly in nonvirtual subsystems when there is a direct feedback connection (in Simulink 4.0 and prior, this wasn't the case). It also means that models compile (update diagram) faster. As a side effect, the execution order for S-functions that incorrectly set the direct feedthrough flag differs from that used in previous versions of Simulink. Consequently, models that contain invalid S-functions may produce different answers in this version of Simulink.

## Added Latched Triggered Subsystems

Now triggered subsystems enable you to implement software triggering, hardware triggering, or a combination of the two. Software triggering is defined as

```
if (trigger_signal_edge_detected) {
   out(t) = f(in(t));
}
```

Hardware triggering is defined as

```
if (trigger_signal_edge_detected) {
   out(t) = f(in(t-h));   // h == last step size
}
```

Previous to this version, triggered subsystems provided software triggering and a form of hardware triggering when a cycle involving triggered subsystems existed. Now, you must explicitly specify whether or not you'd like software or hardware triggering. This is done by selecting `'Latch (buffer) input'` on the Inport blocks in a triggered subsystem.

Each input port of a triggered subsystem configures whether or not the input should be latched. A latched input provides the hardware-triggering semantics for that input port. Type `sl_subsys_semantics` at the MATLAB prompt for more information.

## Self-Triggering Subsystems Are No Longer Allowed

Before this version, you could define the output of a triggered subsystem to directly feed back into the trigger port of the subsystem (with potentially other

additive signals). This resulted in an implicit delay. Now you must explicitly define the delay by inserting a memory block.

## Improved Invalid Model Configuration Diagnostics

This version of Simulink does a better job of detecting and flagging invalid modeling constructs in Simulink models. Consequently models that ran in previous versions of Simulink (sometimes producing incorrect results) may not run in the current release. The changes include:

- Direct feedthrough compensation no longer occurs by default for S-functions (see “Direct Feedthrough Compensation Deprecated” on page 3-13).
- S-functions are now sorted like built-in blocks (see “S-Functions Sorted Like Built-In Blocks” on page 3-14).
- Simulink no longer inserts implicit latches in triggered subsystems that directly or indirectly trigger themselves (see “Self-Triggering Subsystems Are No Longer Allowed” on page 3-14, above). Instead it signals an error when it detects a triggered subsystem loop with unlatched inputs. To avoid the error, you must select the **Latch** option on the triggered subsystem's input ports.
- Simulink now signals an error when it detects invalid configurations of function-call subsystems. See the Subsystem Examples block in the Subsystems library for examples of illegal modeling constructs involving function-call subsystems. You can disable this diagnostic by setting the `Invalid FcnCall Connection` parameter on the **Diagnostics** pane of the **Simulation Parameters** dialog box to `none` or `warning`.

# Known Software Problems

This section updates the Simulink 4.1 documentation set, reflecting known Simulink 4.1 software problems.

## Accelerator Mode Does Not Support Inline Parameters

Accelerator mode does not support the **Inline parameters** option on the **Advanced** pane of the **Simulation Parameters** dialog box. Block outputs may appear at different times in accelerated mode than they do when the model is running in normal mode with the **Inline parameters** option selected.

## Turn the New Wrap Lines Option Off

The MATLAB Command Window has a new **Wrap lines** option. Many of Simulink's error messages are very long. This can cause some display problems. Therefore, when using Simulink, you should turn the **Wrap lines** option off using the **Preferences** setting. For more information on this issue, see the Technical Support Solution 29082 from the MathWorks Web page.

4

# Stateflow 4.1 Release Notes


**New Features** . . . . . 4-2
Smart Transitions . . . . . 4-2
Search & Replace Tool Enhancements . . . . . 4-3
Stateflow Chart Notes . . . . . 4-3
Model Coverage Tool . . . . . 4-3

**Major Bug Fixes** . . . . . 4-4
Editing Crossing Transitions out of Grouped Subcharts . . . 4-4
Disabled and Restored Library Chart Links . . . . . 4-4
Too Many Action Statements During Simulation . . . . . 4-4
False State Inconsistency Runtime Error . . . . . 4-4
MATLAB Variables Improperly Overwritten . . . . . 4-4
Target Options Fields Overwritten . . . . . 4-5
Transitions Assertions . . . . . 4-5
Build Failures with Custom Code . . . . . 4-5

**Upgrading from an Earlier Release** . . . . . 4-6
Transition Actions into Junctions Disallowed . . . . . 4-6

# New Features

This section introduces new features and enhancements added in Stateflow 4.1 (Release 12.1) since the Web-downloadable release of Stateflow 4.0.2 (Release 12).

For information about the features of Stateflow 4.1 added since the release of Stateflow 4.0 (Release 12) and prior versions, see the *What's New* link from the Stateflow product page at `www.mathworks.com`.

---

**Note** For information on new features in Stateflow Coder 4.1, software that generates code for your Stateflow diagrams, see the Stateflow Coder 4.1 Release Notes.

---

## Smart Transitions

Stateflow charts now include the graphical innovation of smart transitions whose ends slide around the surfaces of states and junctions. When the source and/or destination objects are moved and resized in the Stateflow chart, these transitions use sliding and other behaviors to enable users to produce an aesthetically pleasing diagram. See the section “Using Smart Transitions” in Stateflow Help.

---

**Note** Transitions are automatically created with smart behavior on the assumption that this behavior is almost always desirable. However, *self-loop* transitions, must be created without smart transition behavior. See the section “Creating Self-Loop Transitions” in online Help for instructions.

---

## Search & Replace Tool Enhancements

Stateflow's Search & Replace tool has been modified with the following enhancements:

- Regular expression tokenized replacements

  Allows you to dynamically choose replacement text based on matched text.

- Case insensitivity/case preservation

  Replaces text with different sensitivities to the use of upper or lower case characters in the found text.

- Addition of a **Search in:** field

  Now you can select any Stateflow chart in your Simulink model or select the entire Stateflow machine.

See the section "Searching and Replacing in Charts" in Chapter 6, "Exploring and Searching Charts" in the Stateflow documentation for a full description.

## Stateflow Chart Notes

You can now enter annotations to your Stateflow charts that are similar to annotations in Simulink. See the section "Entering Chart Notes" in Stateflow Help.

## Model Coverage Tool

The Simulink Model Coverage tool has been modified to perform model coverage calculations for decisions and conditions of decision in the Stateflow machine and its charts, states, and transitions. This includes Condition and MCDC coverage. See the section "Stateflow Chart Model Coverage" of Chapter 10, "Debugging and Testing" in the Stateflow documentation.

# Major Bug Fixes

The following is a list of known bugs from previous Stateflow version 4.0.2 (Release 12.0) which are *fixed* in Stateflow 4.1. For information about bugs fixed for prior versions, see the online topic "What's New in Stateflow."

---

**Note** For information on major bugs fixed in Stateflow Coder 4.1, software that generates code for your Stateflow diagrams, see the chapter on the Stateflow Coder 4.1 in the *Release Notes*.

---

## Editing Crossing Transitions out of Grouped Subcharts

Editing crossing transitions out of grouped subcharts caused model corruptions in Stateflow versions 3.0 through 4.0.2.

## Disabled and Restored Library Chart Links

Stateflow library chart links that are disabled and then restored caused model corruptions in Stateflow versions 3.0 through 4.0.2.

## Too Many Action Statements During Simulation

Stateflow Debugger caused an error during simulation when a state has more than 255 action statements in Stateflow versions 1.0 through 4.0.2.

## False State Inconsistency Runtime Error

Charts within a Simulink enabled subsystem produced a false state inconsistency runtime error when the subsystem was disabled in Stateflow versions 2.0 through 4.0.3.

## MATLAB Variables Improperly Overwritten

Selecting **Save final value to base workspace** in the properties dialog for a data item caused unrelated MATLAB variables in the workspace to be overwritten in Stateflow versions 4.0 through 4.0.3.

## Target Options Fields Overwritten

The **Custom Initialization** and **Custom Termination** fields in the **Target Options** dialog box were overwritten by empty strings in the data dictionary in Stateflow 4.0.3.

## Transitions Assertions

Transitions containing a temporal trigger combined with other event triggers, such as `after(3E)|E2|E3`, caused assertion errors during parsing in Stateflow 4.0.3.

## Build Failures with Custom Code

When including custom code, build failures occurred due to DOS shell command line length limitations because all user-defined directories on the MATLAB path were added to the include directory list in Stateflow 4.0.3.

# Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from Stateflow 4.0.2 (Release 12.0) or earlier to Stateflow 4.1 (Release 12.1).

## Transition Actions into Junctions Disallowed

Transition actions are now permitted only on transitions that terminate on states. For the following reasons, transition actions are no longer permitted on transitions that terminate on junctions; condition actions should be used instead:

- The semantics of transition actions for transitions into junctions are complex and easily misunderstood and misused.
- The complex semantics of these transition actions also result in generated code that is inefficient. Eliminating these transition actions not only simplifies Stateflow diagrams but also results in generated code that is much more efficient.

If your current model has these transition actions, they are flagged with an error that refers you to Appendix D of the online *Stateflow User's Guide*, "Obsolete Features," for a workaround. In most cases, replacing these transition actions with condition actions gives identical chart behavior.

5

# Real-Time Workshop 4.1 Release Notes


**Release Summary** . . . . . 5-2

**New Features** . . . . . 5-3

**Bug Fixes** . . . . . 5-11

**Platform Limitations for HP and IBM** . . . . . 5-15

**Upgrading from an Earlier Release** . . . . . 5-16

**Known Software Problem** . . . . . 5-19

## Release Summary

Real-Time Workshop 4.1 includes significant new and enhanced features and many improvements in the quality of generated code, including:

- Expression folding, which increases code efficiency and decreases code usage
- External mode support for inlined parameters
- Block states can now be interfaced to externally written code, in a manner similar to signals
- New debugger for Target Language Compiler (TLC) programs
- Support for new Simulink blocks, including control flow constructs such as do-while, for, and if
- Numerous bug fixes

# New Features

This section introduces the new features and enhancements added in the Real-Time Workshop 4.1 since the Real-Time Workshop 4.0.

For information about Real-Time Workshop features that are incorporated from recent releases, see the *Release Notes for Release 12*.

---

**Note** For information about closely related products that extend the Real-Time Workshop, see the sections about the Real-Time Workshop Ada Coder, Real-Time Workshop Embedded Coder, and xPC Target.

---

## Block Reduction Option On by Default

The **Block reduction** option (on the Advanced page of the **Simulation Parameters** dialog box) is now turned on by default. In prior releases, this option was off by default.

**Block reduction** collapses certain groups of blocks into a single, more efficient block, or removes them entirely. This results in faster model execution during simulation and in generated code.

See "Block Reduction Option" in the *Real-Time Workshop User's Guide* for further information.

## Buffer Reuse Code Generation Option

The **Buffer reuse** option is now available via the **General Code Generation Options (cont.)** category of the Real-Time Workshop page. When the **Buffer reuse** option is selected, signal storage is reused whenever possible.

In previous releases, this option was available only through MATLAB `set_param` and `get_param` commands, such as:

```
set_param(gcs,'bufferreuse','on')
```

The `set_param` and `get_param` commands are still supported.

See "Buffer Reuse Option" and "Signals: Storage, Optimization, and Interfacing" in the *Real-Time Workshop User's Guide* for further information.

## Build Directory Validation

The build process now disallows building programs in the MATLAB directory tree. If you attempt to generate code in the MATLAB directory tree, an error message will be displayed prompting you to change to a working directory that is not in the MATLAB directory tree. On a PC, you can continue to build in the directory *matlabroot*/Work.

The build process also prevents building programs when *matlabroot* has a dollar sign ($) in its MATLAB directory name.

## Build Subsystem Enhancements

The **Build Subsystem** feature, introduced in Real-Time Workshop 4.0, lets you generate code and build an executable from any nonvirtual subsystem within a model. In Real-Time Workshop 4.1, the Build Subsystem feature has been enhanced as follows:

- The **Build Subsystem** window now displays additional information about block parameters referenced by the subsystem.
- From the **Build Subsystem** window, you can now inline any parameter or set the storage class of any parameter.

See "Generating Code and Executables from Subsystems" in the *Real-Time Workshop User's Guide* for further information.

## C API for Parameter Tuning Documented

The Real-Time Workshop provides data structures and a C API that enable a running program to access model parameters without use of external mode. This API has now been fully documented.

To access model parameters via the C API, you generate a model-specific parameter mapping file, model_pt.c. This file contains parameter mapping arrays containing information required for parameter tuning.

See "C API for Parameter Tuning" in the *Real-Time Workshop User's Guide* for information on how to generate and use the parameter mapping file.

## Code Readability Improvements

Improvements to the readability of generated code include:

- Elimination of redundant parentheses.
- Long C statements in the generated code are now split across multiple lines.
- Block comments are more informative.

## Control Flow Blocks Support

Simulink 4.1 implements a number of blocks that support logic constructs such as if-else and switch, and looping constructs such as do-while, for, and while. The Real-Time Workshop 4.1 supports code generation from these blocks.

For further information on the flow control blocks, see "Control Flow Statements" in *Using Simulink*.

## Expression Folding

Expression folding is a code optimization technique that minimizes the computation of intermediate results at block outputs, and the storage of such results in temporary buffers or variables. Wherever possible, the Real-Time Workshop collapses, or "folds," block computations into single expressions, instead of generating separate code statements and storage declarations for each block in the model.

Expression folding dramatically improves the efficiency of generated code, frequently achieving results that compare favorably to hand-optimized code. In many cases, model computations fold into a single highly optimized line of code.

Most Simulink blocks support expression folding.

For further information, see "Expression Folding" in the *Real-Time Workshop User's Guide*.

## External Mode Enhancements

### Inline Parameters Support

The Real-Time Workshop now lets you use the **Inline parameters** code generation option when building an external mode target program. When you inline parameters, you can use the **Model Parameter Configuration** dialog to remove individual parameters from inlining and declare them to be tunable. This allows you to improve overall efficiency by inlining most parameters, while at the same time retaining the flexibility of run-time tuning for selected parameters that are important to your application. In addition, the **Model Parameter Configuration** dialog offers you options for controlling how parameters are represented in the generated code.

Each time Simulink connects to a target program that was generated with **Inline parameters** on, the target program uploads the current value of its tunable parameters (if any) to the host. These values are assigned to the corresponding MATLAB workspace variables. This procedure ensures that the host and target are synchronized with respect to parameter values.

All targets that support external mode (i.e., `grt`, `grt_malloc`, and Tornado) now allow inline parameters.

See "Overview of External Mode Communications" in the *Real-Time Workshop User's Guide* for further information.

### Status Bar Display

When Simulink is connected to a running external mode target program, the simulation time and other status bar information is now displayed and updated just as it would be in normal mode.

## Generate Comments Option

This option lets you control whether or not comments are written in the generated code. See "Generate Comments" in the *Real-Time Workshop User's Guide* for further information.

## Include System Hierarchy in Identifiers

When this option is on, the Real-Time Workshop inserts system identification tags in the generated code (in addition to tags included in comments). The tags help you to identify the nesting level, within your source model, of the block that generated a given line of code.

See "Include System Hierarchy in Identifiers" in the *Real-Time Workshop User's Guide* for further information.

## Rapid Simulation Target Supports Inline Parameters

The Rapid Simulation Target now works with **Inline parameters** on. Note that when **Inline parameters** is on, the storage class for all parameters and signals is silently forced to auto.

## S-Function Target Enhancements

The S-Function Target **Generate S-function** feature, introduced in Real-Time Workshop 4.0, lets you generate an S-function from a subsystem. This feature has been enhanced as follows:

- The **Generate S-function** window now displays additional information about block parameters referenced by the generating subsystem.
- If you have installed and licensed the Real-Time Workshop Embedded Coder, the **Generate S-function** window lets you invoke the Embedded Coder to generate an S-function wrapper.

See "Automated S-Function Generation" in the *Real-Time Workshop User's Guide* for details.

## Storage Classes for Block States

For certain block types, the Real-Time Workshop lets you control how block states in your model are stored and represented in the generated code. Using the **State Properties** dialog, you can:

- Control whether or not states declared in generated code are interfaceable (visible) to externally written code. You can also specify that signals are to be stored in locations declared by externally written code.
- Assign symbolic names to block states in generated code.

For further information, see "Block States: Storing and Interfacing" in the *Real-Time Workshop User's Guide*.

## Support for tilde (~) in Filenames on UNIX Platforms

All filename fields in Simulink now support the mapping of the tilde (~) character in filenames. For example, in a To File block you can specify `<code>~/outdir/file.mat</code>`. On most systems, this will expand to `/home/$USER/outdir/file.mat`. The Real-Time Workshop uses the expanded names.

## Target Language Compiler 4.1

This section summarizes changes that have been made to the Target Language Compiler in this release. See also "TLC Compatibility Issues" on page 5-17.

### New TLC Debugger

The TLC debugger helps you identify programming errors in your TLC code. The debugger lets you set breakpoints in your TLC code, execute TLC code line-by-line, examine and change variables, and perform many other useful operations.

The TLC debugger operates during code generation, incurring almost no overhead in the code generation process. You can invoke the debugger:

- By selecting options in the **TLC debugging options** category of the Real-Time Workshop page
- By including `%breakpoint` statements in your TLC file.
- By using the MATLAB `tlc` command, as in

  ```
  tlc -dc <options>
  ```

- By using the `-dc` build option in the **System target file** field of the **Real-Time Workshop** page.

For further information, see "Debugging TLC" in the *Target Language Compiler Reference Guide*.

### model.rtw File Format Changes

The format of the *`model`*`.rtw` file has changed. See "TLC Compatibility Issues" on page 5-17.

### Cleanup of Block I/O Connection Handling in TLC

The handling of signal connections in `rtw/c/tlc/blkiolib.tlc` and `rtw/ada/tlc/blkiolib.tlc` was reworked. See the discussion of `LibBlockInputSignal` in "TLC Function Library Reference" in the *Target Language Compiler Reference Guide*.

### Literal String Support

If a string constant is prefixed by the `L` format specifier, then no escape character processing is performed on that string. This is useful for specifying PC style path or directory names, as in this example.

```
%addincludepath L"C:\mytlc"
```

### New TLC Library Functions

The following functions have been added to the TLC Function Library:

- `LibBlockInputSignalConnected`
- `LibBlockInputSignalLocalSampleTimeIndex`
- `LibBlockInputSignalOffsetTime`
- `LibBlockInputSignalSampleTime`
- `LibBlockInputSignalSampleTimeIndex`
- `LibBlockOutputSignalOffsetTime`
- `LibBlockOutputSignalSampleTime`
- `LibBlockOutputSignalSampleTimeIndex`
- `LibBlockMatrixParameterBaseAddr`
- `LibBlockParameterBaseAddr`
- `LibBlockNonSampledZC`

  See "TLC Function Library Reference" in the *Target Language Compiler Reference Guide* for information on these functions.

### TLC Bug Fixes

- Fixed a bug where local variables of calling functions were sometimes incorrectly visible to called functions.
- The `ISINF`, `ISNAN`, and `ISFINITE` functions now work for complex values.
- The `%filescope` directive now works as documented.

- Zero indexing on complex numbers is now supported.

  In prior releases, the Target Language Compiler allowed 0 indexing for integer and real values, but not for complex values. This restriction has been removed in the Target Language Compiler 4.1, as shown in the following example.

  ```
  %assign a = 1.0 + 3.0i
  %assign b = a[0] %% zero index now allowed
  ```

- Fixed a crash that occurred if `ROLL_ITERATIONS` was called outside of a `%roll` construct. `ROLL_ITERATIONS` returns `NULL` if called outside of a `%roll` construct.
- TLC now allows use of any path separator character independent of operating system. You can use either \ or / as a path separator character on Unix or Windows).
- Fixed a bug in the compare for equality operation. `0.0` now compares equal to `-0.0`.

# Bug Fixes

The Real-Time Workshop 4.1 includes the bug fixes described in this section. See also "TLC Bug Fixes" on page 5-9 for bug fixes specific to the Target Language Compiler.

## Block Reduction Crash Fixed

A problem that crashed MATLAB due to a segmentation fault during the block reduction process has been fixed. This problem occurred only if the **Block Reduction** option was on, and if a Scope block was connected to a block that was removed due to block reduction.

## Build Subsystem Gives Better Error Message for Function Call Subsystems

The **Build Subsystem** feature does not currently support triggered function-call subsystems. The Real-Time Workshop now gives a a more informative error message when a **Build Subsystem** is attempted with a triggered function-call subsystem.

## Check Consistency of Parameter Storage Class and Type Qualifier

The Real-Time Workshop now checks for consistency of parameter storage class and type qualifier when a parameter is specified by both the **Model Parameter Configuration** dialog box and a referenced Simulink data object.

## Code Optimization for Unsigned Saturation and DeadZone Blocks

When the lower limit of a Saturation or DeadZone block is a zero and is nontunable, and the data type is unsigned, the comparison against the lower limit is omitted from the code. Similarly, if the upper or lower limit of the Saturation block is nontunable and nonfinite, the comparison against the infinite limit is omitted.

## Correct Code Generation of Fixed-Point Blockset Blocks in DSP Blockset Models

A code generation bug involving some DSP Blockset blocks (see list below) was fixed. When these blocks were driven by a block from the Fixed-Point Blockset, generated code would write outside array memory bounds. The following DSP Blockset blocks generated incorrect code:

- Delay Line
- Frame Status Conversion
- Matrix Multiply
- Multiport Selector
- Pad
- Submatrix
- Window Function
- Zero Pad

## Correct Compilation with Green Hills and DDI Compilers

Compilation errors for files associated with matrix inversion in the *matlabroot*`/rtw/c/libsrc` directory were fixed. These errors occurred with the Green Hills and DDI compilers.

## Fixed Build Error with Models Having Names Identical to Windows NT Commands

This fix prevents an error that occurred when building models having names identical to Windows NT internal commands. Examples would be models named `verify` or `path`. Such model names are now allowed.

## Fixed Error Copying Custom Code Blocks

An error in the Custom Code block `Copyfcn` callback was fixed. The problem caused an error when copying a custom code block within a model.

## Fixed Error in commonmaplib.tlc

A typo in rev 1.17 of `commonmap.tlc` was fixed. This typo caused an error during code generation, when using the `grt_malloc` target with **External mode** selected.

## Fixed Name Clashes with Run-Time Library Functions

The Real-Time Workshop now uses the macros `rt_min` and `rt_max` to avoid clashing with run-time library `min` and `max` functions.

## Improved Handling of Sample Times

The sample time handling for the S-function and ERT targets has been improved to use the compiled sample time instead of the user specified sample time on the input port blocks.

## Look-Up Table (n-D) Code Generation Bug Fix

The Real-Time Workshop now generates correct code for Look-Up Table (n-D) blocks having 5 or more dimensions with different dimension sizes.

## Parenthesize Negative Numerics in Fcn Block Expressions

Fcn block expressions in the generated code failed to compile in the case of a unary operator preceding a workspace variable with a negative value, such as the expression

```
-v*u
```

Such expressions are now enclosed in parentheses, as in

```
(-v) * u
```

## Removed Unnecessary Warnings and Declarations from Generated Code

Several unnecessary warnings and declarations in the generated code have been removed. These include:

- In functions where the `tid` argument is not referenced, the declaration

  ```
  (void)tid
  ```

  is no longer generated. (The `tid` argument is required because the function API is predefined.)
- Warnings involving `const` casts were suppressed in some of the `rtw/c/libsrc` modules.

## Retain .rtw File Option Now Works in Accelerator Mode

In previous releases, the **Retain .rtw file** option (on the TLC Debugging Options page of the **Simulation Parameters** dialog box) was ignored if Simulink was in Accelerator mode. Now, you can retain the *`model`*`.rtw` file during a build, regardless of the simulation mode.

## S-Function Target Memory Allocation Bug Fix

A segmentation fault during generation of S-functions was removed by fixing the memory management of the port data structure.

# Platform Limitations for HP and IBM

**Note** The Release 12.0 platform limitations for Real-Time Workshop for the HP and IBM platforms still apply to Release 12.1. That limitation is described below.

## Tunable Parameters Dialog Box

On the HP and IBM platforms, the Real-Time Workshop opens the Release 11 **Tunable Parameters** dialog box in place of the **Model Parameter Configuration** dialog box. Although they differ in appearance, both dialogs present the same information and support the same functionality.

# Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving from the Real-Time Workshop 4.0 (Release 12.0) to the Real-Time Workshop 4.1.

For information about upgrading from a release earlier than 4.0, see the *Release Notes for Release 12*.

## RTWInfo Property Changes

If you use the Simulink Data Object classes `Simulink.Signal` or `Simulink.Parameter`, or have implemented classes derived from these, note the following information concerning the `RTWInfo` properties.

In Release 12.0, the `RTWInfo` class had a `TypeQualifier` property, corresponding to the **RTW storage type qualifier** field of signal ports and parameters.

Real-Time Workshop 4.1 now supports creation of custom storage classes, removing the need for the `TypeQualifier` property. We recommend that you use custom storage classes when type qualification is needed. See “Custom Storage Classes for Data Objects” on page 26-3 for further information.

By default, the `TypeQualifier` property of `RTWInfo` objects is no longer visible in the Simulink Data Explorer. Also, the `TypeQualifier` property is no longer written to `ObjectProperties` records in the *`model`*`.rtw` file. For backward compatibility, the `TypeQualifier` property remains active. The property can be set and retrieved through a direct reference. For example,

```
Kp.RTWInfo.TypeQualifier = 'const'
```

or

```
tq = Kp.RTWInfo.TypeQualifier
```

You can make the `TypeQualifier` property visible in the Simulink Data Explorer for the duration of a MATLAB session. To do this, execute the following command prior to opening the Simulink Data Explorer),

```
feature('RTWInfoTypeQualifier',1)
```

The above command also directs the Real-Time Workshop to include the `TypeQualifier` property in `ObjectProperties` records in the *`model`*`.rtw` file.

For further information see “Simulink Data Objects and Code Generation” in the *Real-Time Workshop User's Guide.*

## S-Function Target MEX-Files Must Be Rebuilt

S-function MEX-files generated by the S-function target under Release 11 are not compatible with Release 12. The incompatibilities are due to new features, such as parameter pooling, introduced in Release 12.0.

If you have built S-function MEX-files with the S-function target under Release 11, you must rebuild them. See “The S-Function Target” in the *Real-Time Workshop User's Guide* for further information.

## TLC Compatibility Issues

### model.rtw File Format Changes

The format of the *model*`.rtw` file has changed. For further information, see “model.rtw Changes Between Real-Time Workshop 4.0 and 4.1” in the *Target Language Compiler Reference Guide.*

### Reordering of BlockTypeSetup and BlockInstanceSetup Calls

During the initialization phase of code generation, the Target Language Compiler makes a pass over all blocks in the model and executes several functions, including:

- Each block's `BlockTypeSetup` function the first time that block type is encountered.
- Each block's `BlockInstanceSetup` function. `BlockInstanceSetup` is called for all instances of a given block type in the model.

The order in which these calls are made is significant, because the `BlockInstanceSetup` function may depend upon global variables that are initialized by the `BlockTypeSetup` function.

In Release 12.1, the `BlockTypeSetup` function is called before the `BlockInstanceSetup`. This corrects a problem in previous releases, where `BlockInstanceSetup` was erroneously called first. You may need to change your S-functions or block implementations if they depend upon the previous behavior.

### Obsolete Code Generation Variables

The code generation variables `FunctionInlineType` and `PragmaInlineString` are now obsolete. These variables controlled the generation of inlined functions. In the current release, you can generate inlined functions from subsystems, as described in "Nonvirtual Subsystem Code Generation" in the *Real-Time Workshop User's Guide*.

# Known Software Problem

This section updates the Real-Time Workshop 4.1 documentation set, reflecting a known software problem.

## Turn the New Wrap Lines Option Off

The MATLAB Command Window has a new **Wrap lines** option. Real-Time Workshop frequently displays very long message lines as a build progresses. This can cause some display problems. Therefore, when using Real-Time Workshop, you should turn the **Wrap lines** option off using the **Preferences** setting. For more information on this issue, see the Technical Support Solution 29082 from the MathWorks Web page.

6

# CDMA Reference Blockset 1.1 Release Notes

**New Features** . . . . . 6-2
Frame-Based Signal Support . . . . . 6-2

**Upgrading from an Earlier Release** . . . . . 6-3

# New Features

This section introduces the new features added in the CDMA Reference Blockset 1.1 since the CDMA Reference Blockset 1.0.2 (Release 12).

## Frame-Based Signal Support

The CDMA Reference Blockset 1.1 supports frame-based signals and matrix signals. As a result, it is no longer necessary to change the signal attributes of blocks from the Communications Blockset or the DSP Blockset in order to use them in models with blocks from the CDMA Reference Blockset. This makes it even easier to use the CDMA Reference Blockset with those blocksets.

All of the blocks in Version 1.1 allow both frame-based and sample-based signals, with the exception of the following blocks, which allow only frame-based signals:

- IS-95A Fwd Ch Base Station Transmitter Interface
- IS-95A Fwd Ch Descrambler
- IS-95A Fwd Ch Detector
- IS95A Long Code Generator
- IS-95A Rev Ch Detector
- IS-95A Rev Ch Rake Finger
- IS-95 A Rev Ch Walsh Modulation and Spreading
- IS95A Short Code Generator
- IS95A Walsh Code Generator

You can open the new Version 1.1 libraries by typing `cdmalibv1p1` at the MATLAB prompt, or by using the Simulink Library Browser.

# Upgrading from an Earlier Release

If you are upgrading from the CDMA Reference Blockset Version 1.0.2 to Version 1.1, models built with Version 1.0.2 will continue to work in Version 1.1.

However, you may encounter problems if you combine old blocks from Version 1.0.2 in a model with the following new Version 1.1 code generators:

- IS95A Long Code Generator
- IS95A Short Code Generator
- IS95A Walsh Code Generator

The Version 1.0.2 blocks that do not support frames, and so cannot be used with the Version 1.1 code generators, are:

- IS-95A Fwd Ch Base Station Transmitter Interface
- IS-95A Fwd Ch Descrambler
- IS-95A Fwd Ch Detector
- IS-95A Rev Ch Detector
- IS-95A Rev Ch Rake Finger
- IS-95 A Rev Ch Walsh Modulation and Spreading

If you continue to add blocks to a model built with Version 1.0.2, we recommend that you use blocks from the Version 1.0.2 libraries. You can open the Version 1.0.2 libraries by typing `cdmalib` at the MATLAB prompt.

7

# Communications Blockset 2.0.1 Release Notes


**New Features** . . . . . 7-2
Setting Simulink Preferences Automatically . . . . . 7-2
Converting Between Bipolar and Unipolar Signals . . . . . 7-2
Choosing Seeds for Random-Output Blocks . . . . . 7-2
Using Error Counts to Control Simulation Duration . . . . 7-3
Choosing the Algorithm for Integrator Blocks . . . . . 7-3

**Major Bug Fixes** . . . . . 7-4
Bug Fixes Incorporated from Release 12.0 . . . . . 7-4

**Upgrading from an Earlier Release** . . . . . 7-5

**Known Software and Documentation Problems** . . . . 7-6
Code Generation Limitations . . . . . 7-6
Limited Frame and Matrix Support . . . . . 7-6
Errors in Printed Version of Documentation . . . . . 7-7

# New Features

**Note** The Communications Blockset was a new product that was introduced with Release 12.0. The Communications Blockset incorporates the functionality of the blocks that were included in the Communications Toolbox 1.4 (Release 11), with the addition of the new features summarized below. The Communications Toolbox is described in the *Release Notes for Release 12*.

This section introduces the new features and enhancements added in the Communications Blockset 2.0.1 since the Communications Blockset 2.0 (Release 12.0).

For information about Communications Blockset features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## Setting Simulink Preferences Automatically

The new `commstartup.m` script sets certain Simulink preferences to values that are most appropriate for the simulation of communication systems. To use this script, type the command `commstartup` in your `startup.m` file or in the MATLAB Command Window.

## Converting Between Bipolar and Unipolar Signals

The Utility Functions library contains new blocks that convert between bipolar and unipolar signals. The blocks are Bipolar to Unipolar Converter and Unipolar to Bipolar Converter.

## Choosing Seeds for Random-Output Blocks

The `randseed` function is a new function that generates prime numbers for use as **Initial seed** parameters in blocks that produce random output. Compared to composite seeds, prime seeds yield output that has better statistical properties.

## Using Error Counts to Control Simulation Duration

You can now configure the Error Rate Calculation block so that it automatically stops the simulation upon detecting a specified number of errors. You do not need to know in advance how long it will take to accumulate that many errors.

## Choosing the Algorithm for Integrator Blocks

The Discrete Modulo Integrator block now allows you to choose the integration method using a mask parameter. The corresponding mask parameter in the Windowed Integrator block has changed its name from **Method** to **Integration method** for consistency with other integration blocks.

# Major Bug Fixes

The Communications Blockset 2.0.1 includes several bug fixes, including the following:

- The M-FSK Baseband Modulator, M-FSK Baseband Demodulator, M-FSK Passband Modulator, and M-FSK Passband Demodulator blocks now use the correct tone spacing.
- The PN Sequence Generator block now generates only binary values, and the numbers in the sequence do not depend on the frame status or size.

## Bug Fixes Incorporated from Release 12.0

The Communications Blockset 2.0.1 includes several bug fixes that were made in Release 12.0.

# Upgrading from an Earlier Release

If you are upgrading from an earlier release to the Communications Blockset 2.0.1, then note these issues:

- The Binary Symmetric Channel block dialog box now omits the **Input vector length** and **Sample time** parameters because the block now determines these quantities automatically. However, if you open a model in Release 12.1 that contains the Release 12.0 Binary Symmetric Channel block, then the Command Window might display warnings about block parameters. To suppress these warnings in the future, simply save the model from Release 12.1.
- In the Communications Blockset 2.0.1, any model that includes a digital passband modulator block or a digital passband demodulator block must use a variable-step solver rather than a fixed-step solver. To configure a model so that it uses a variable-step solver, select **Simulation parameters** from the model window's **Simulation** menu and then set the **Type** parameter on the **Solver** panel to **Variable-step**.

See the *Release Notes for Release 12* for upgrade issues involved in moving from the Communications Toolbox 1.4 (Release 11) to the Communications Blockset 2.0.1.

# Known Software and Documentation Problems

This section updates the Communications Blockset 2.0.1 documentation set, reflecting known Communications Blockset 2.0.1 software and documentation problems.

## Code Generation Limitations

Several Communications Blockset blocks are incompatible with Real-Time Workshop. As a result, Real-Time Workshop cannot generate code for models that include these blocks:

- Continuous-Time Eye and Scatter Diagrams block
- Multipath Rayleigh Fading Channel block
- Rician Fading Channel block
- Triggered Read from File block
- Triggered Write to File block
- Voltage-Controlled Oscillator block
- Blocks in the Analog Passband and Digital Passband sublibraries of the Modulation library
- Blocks in the CPM sublibrary of the Digital Baseband sublibrary of the Modulation library

## Limited Frame and Matrix Support

The Communications Blockset provides limited support for matrix and frame-based signals. In a future release, more blocks will support multichannel behavior, and more blocks will be optimized for faster frame-based processing. Release 12.1 strives to be forward-compatible in the sense that future signal support modes should not invalidate current modes and should minimize the difficulty of upgrading from Release 12.1 to a future release.

As a consequence of this view, some blocks now use strict guidelines to determine the kinds of signals that they accept. One consideration is that if a block will ultimately support frame-based multichannel signals, then a sample-based vector input might potentially represent either a frame of data from a single channel or a set of samples from multiple channels. Therefore, even if such a block does not currently provide such comprehensive signal

support, it accepts only frame-based vectors, whose interpretation is unambiguous.

## Errors in Printed Version of Documentation

The printed version of the *Communications Blockset User's Guide* contains these errors:

- In the example, "Passband Digital Modulation," the instructions on page 2-82 for configuring the Mean block should say, "Set **Reset port** to **None**" instead of "Uncheck the **Reset port** check box." The electronic version of the instructions is correct.
- Reference pages for some blocks in the CPM modulation libraries omit the **Symbol prehistory** and/or **Phase offset** parameters. The electronic versions of the reference pages are correct. The affected blocks are:
  - CPM Demodulator Baseband
  - GMSK Demodulator Baseband
  - MSK Demodulator Baseband
  - CPFSK Demodulator Baseband
  - CPM Demodulator Passband
  - GMSK Demodulator Passband

8

# Control System Toolbox 5.1 Release Notes

**New Features** . . . . . 8-2
SISO Design Tool Enhancements . . . . . 8-2
Algorithmic Enhancements . . . . . 8-3

**Platform Limitations for HP and IBM** . . . . . 8-4
Preference and Property Editors . . . . . 8-4
SISO Design Tool . . . . . 8-4

# New Features

This section introduces the new features and enhancements added in the Control System Toolbox 5.1 since the Control System Toolbox 5.0 (Release 12.0).

For information about Control System Toolbox features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## SISO Design Tool Enhancements

Enhancements to the SISO Design Tool include the following:

- Save/load session — You can now save the state of the SISO Design Tool before exiting and restore it at a later time.
- Graphical design constraints — You can use a new set of design constraints that you move and shape with the mouse or specify by typing their parameter values. For example, the design constraints for root locus include damping ratios, settling time, natural frequency, and percent overshoot.
- Interactive open-loop Nichols plot — You can use a new view to graphically tune the parameters of the feedback compensator. The open-loop Nichols plot contains the same functionality as the open-loop Bode diagram and root locus.
- Interactive Prefilter Design view — You can now interactively shape the prefilter compensator. Both interactive graphics and a right-click menu are available.
- New compensator format — You can display compensator transfer functions in the natural frequency format, which is of the form

$$dcgain \times \frac{(1 + s/\omega_{z1})}{(1 + s/\omega_{p1})} \ldots$$

  where $\omega_{z1}$,$\omega_{z2}$, ... and $\omega_{p1}$, $\omega_{p2}$, ..., are the natural frequencies of the zeros and poles, respectively.

For detailed examples of how to use the SISO Design Tool, see “Designing Compensators” in *Getting Started with the Control System Toolbox*.

## Algorithmic Enhancements

Enhancements to controls algorithms include:

- M-circle grid for Nyquist plots
- A new function, `bandwidth`, that calculates the bandwidth of SISO systems
- LTI models with complex data — You can now specify transfer functions or state-space models with complex coefficients. Most commands work on such models, *except* for the following:
  - Time response commands
  - `rlocus`
  - `margin`, `allmargin`
- New LTI method `conj` for use with LTI models with complex data (works for TF, ZPK, SS objects)

# Platform Limitations for HP and IBM

**Note** The Release 12.0 platform limitations for the HP and IBM platforms still apply to Release 12.1. Those limitations are described below.

## Preference and Property Editors

The Toolbox Preferences Editor and Response Property Editor are not supported. The LTI Viewer Preferences Editor is supported with a reduced set of features (no tools to set preferences for grids, fonts, colors, or phase wrapping).

## SISO Design Tool

The following features are not supported for the SISO Design Tool:

- Preference and property editing
- Compensator format editing
- Storing and retrieving compensators
- Design Constraints

In addition, the SISO Tool Export window has fewer export options.

9

# Data Acquisition Toolbox 2.1 Release Notes


**New Features** . . . . . 9-2
Additional National Instruments Hardware Support . . . . 9-2
High Resolution Sound Card Support . . . . . 9-2

**Upgrading from an Earlier Release** . . . . . 9-3
Events, Callbacks, and Function Handles . . . . . 9-3
Vendor Drivers . . . . . 9-3
Removed and Obsolete Functions . . . . . 9-4
Two daqhwinfo Fields Removed . . . . . 9-4

**Known Software and Documentation Problems** . . . . 9-5
High Resolution Sound Cards . . . . . 9-5
The putdata Function . . . . . 9-5
The daqread Function . . . . . 9-5
Vendor Limitations . . . . . 9-5

# New Features

This section introduces the new features and enhancements added in the Data Acquisition Toolbox 2.1 since the Data Acquisition Toolbox 2.0 (Release 12.0).

For information about Data Acquisition Toolbox features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## Additional National Instruments Hardware Support

These additional National Instruments boards are supported:

- 445x Dynamic Signal Acquisition (DSA) Series boards
- Analog output boards
- Digital I/O boards (no streaming)

## High Resolution Sound Card Support

High resolution (greater than 16-bit) sound cards are supported. You configure the resolution of your sound card with the `BitsPerSample` property. `BitsPerSample` can be `8` or any value between `16` and `32`.

---

**Note** Some sound card drivers may not support this feature because they are still in beta.

---

# Upgrading from an Earlier Release

**Note** The following issue is involved in upgrading from the Data Acquisition Toolbox 2.0 to Data Acquisition Toolbox 2.1.

## Events, Callbacks, and Function Handles

Action properties and action functions are now referred to as callback properties and callback functions. This new terminology is reflected in new names for the associated properties and functions. The general rule for the name changes is to change “Action” to “Fcn” for properties, and “action” to “callback” for functions. For example, `TimerAction` has been renamed `TimerFcn`, and `daqaction` has been renamed `daqcallback`.

Additionally, if you want to automatically pass the object and event information to the callback function, then you must specify the function as either a function handle or as a cell array. Note that you can also specify the callback function as a string. In this case, the callback is evaluated in the MATLAB workspace and no requirements are made on the function's input arguments.

**Note** The following issues are involved in upgrading from the Data Acquisition Toolbox 1.0.1 to the Data Acquisition Toolbox 2.1.

## Vendor Drivers

If you are not using the latest drivers for your hardware, and you experience problems with the toolbox, then you should upgrade your drivers:

- For National Instruments hardware, you should use NI-DAQ 6.7.0 (`http://www.ni.com/nidaq`). Note that you should be able to use NI-DAQ 6.8, although the toolbox has not been fully tested against this driver version.
- For Agilent Technologies hardware, you should use VXI Plug and Play Drivers version A.06.04 (`http://www.tm.agilent.com`).

## Removed and Obsolete Functions

The `nidaq`, `winsound`, and `hpe1432` functions have been removed.

The `obj2code` function has been renamed to `obj2mfile`. `obj2code` will still work but it now produces a warning.

## Two daqhwinfo Fields Removed

The `ConversionExtraScaling` and `ConversionOffset` fields of the `daqhwinfo` function have been removed. You can use the `NativeScaling` and `NativeOffset` properties to convert native data to doubles.

# Known Software and Documentation Problems

This section updates the Data Acquisition Toolbox 2.1 documentation set, describing known software problems.

## High Resolution Sound Cards

To use the high resolution capabilities for some sound cards, you may need to configure `BitsPerSample` to 32 even if your device does not use that number of bits.

## The putdata Function

You should not modify the `BitsPerSample`, `InputRange`, `SensorRange`, and `UnitsRange` properties after calling `putdata`. If these properties are modified, all data is deleted from the data acquisition engine. If you add a channel after calling `putdata`, then all data will be deleted from the buffer.

## The daqread Function

Property values specified as a cell array are not returned from a `.daq` file.

## Vendor Limitations

Limitations associated with the supported vendors are described in the *Release Notes for Release 12*.

10

# Database Toolbox 2.2 Release Notes

**New Features** . . . . . 10-2
Retrieve Data as numeric or structure Format . . . . . 10-2
Export Numeric or Structure Data . . . . . 10-2
Specify Error Handling Preference . . . . . 10-2
LONGCHAR Data Type Supported . . . . . 10-3

**Platform Limitations for HP and Linux** . . . . . 10-4
logintimeout Not Supported on Linux . . . . . 10-4
No HP 10.2 Support . . . . . 10-4

# New Features

This section introduces the new features and enhancements in the Database Toolbox 2.2, added since the Database Toolbox 2.1 (Release 12.0).

For information about Database Toolbox features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## Retrieve Data as numeric or structure Format

You can specify as a preference the format in which data is retrieved from a database, called the `DataReturnFormat`. Options are `numeric`, `structure`, and `cellarray`. Previous versions of the Database Toolbox only supported the `cellarray` format.

You can use the `numeric` format when all data to be retrieved is numeric or when non-numeric data is not relevant. Any strings in the data are converted as specified in the `NullNumberRead` preference, for example, `NaN`. The `numeric` format allows for a significant reduction in memory usage. The `structure` format allows you to access the retrieved data as a MATLAB structure.

To set the preference, use the `setdbprefs` function. In the Visual Query Builder, set the format using the **Query -> Preferences** menu. For details on the options, see the reference page for `setdbprefs`.

## Export Numeric or Structure Data

Using `insert` and `update`, you can export data that is a numeric matrix, a structure, or a cell array. Previous versions only supported data in a cell array for export. There is no specific action you take to specify the format for exported data; the data is exported in its current MATLAB format.

## Specify Error Handling Preference

You can specify the behavior for handling errors when retrieving data from a database, called the `ErrorHandling` preference. The options are `store`, `report`, and `empty`. Previous versions of the Database Toolbox only supported the `store` option, meaning the error message is stored in the returned data. The `report` option immediately displays any error messages in the Command Window. For the `empty` option, the error message is in the returned data, but it returns objects that could not be created as empty handles.

To set the preference, use the `setdbprefs` function. In the Visual Query Builder, set the preference using the **Query -> Preferences** menu. For details on the options, see the reference page for `setdbprefs`.

## LONGCHAR Data Type Supported

The `LONGCHAR` data type is supported. This is called the Memo data type in Microsoft Access.

# Platform Limitations for HP and Linux

## logintimeout Not Supported on Linux

On LINUX platforms, `logintimeout` is not supported.

---

**Note** The Release 12.0 platform limitation for the HP platform still applies to Release 12.1. That limitation is described below.

---

## No HP 10.2 Support

The Database Toolbox is not supported on the Hewlett-Packard 10.2 platform.

11

# Datafeed Toolbox 1.3 Release Notes

**New Features** . . . . . 11-2
Hyperfeed Support Added . . . . . 11-2
Hyperfeed Function Summary . . . . . 11-2

**FT Interactive Data Server** . . . . . 11-3

# New Features

This section introduces the new features and enhancements added in the Datafeed Toolbox 1.3 since Datafeed Toolbox 1.2 (Release 12.0).

For information about Datafeed Toolbox features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## Hyperfeed Support Added

The Datafeed Toolbox Version 1.3 adds the Hyperfeed data server to the list of previously supported servers (i.e., Bloomberg, FT Interactive Data, and Yahoo). Version 1.3 provides functions to obtain price, profile, historical and tick data from Hyperfeed.

## Hyperfeed Function Summary

**Table 11-1: Datafeed Toolbox Hyperfeed Functions**

| Function | Purpose |
|---|---|
| `close` | Close connection |
| `fetch` | Request data |
| `get` | Get connection properties |
| `hyperfeed` | Connect to Hyperfeed |
| `isconnection` | True if valid connection |

# FT Interactive Data Server

The default location for installing FT Interactive Data client software is `C:\RPLUS`.

For successful operation after installation, add this directory to the Windows system path:

- For Windows 95/98, enter this directory in the `autoexec.bat` file.
- For Windows NT, use the **System** Control Panel to add this directory to the path.

12

# Developer's Kit for Texas Instruments™ DSP 1.1 Release Notes


**Introduction to Developer's Kit for Texas Instruments™ DSP** . . . . . 12-2

**New Features** . . . . . 12-4
New Link Functions for CCS IDE . . . . . 12-4
New Interface Between the Developer's Kit and FDATool . . 12-5
Unified CCS IDE and C6701 EVM Targeting . . . . . 12-5
New Run-Time Libraries . . . . . 12-6
Architecture Change in CCSDSP Object . . . . . 12-6
Updated Function writemsg Help Text . . . . . 12-7

**Major Bug Fixes** . . . . . 12-8

**Upgrading from an Earlier Release** . . . . . 12-9
New Libraries at Run-Time . . . . . 12-9
CCSDSP Changes . . . . . 12-9
Unified Targeting . . . . . 12-9

**Known Software and Documentation Problems** . . . 12-10
Operating Systems . . . . . 12-10
Linking . . . . . 12-10
Targeting . . . . . 12-12
Demos . . . . . 12-14
Datatypes . . . . . 12-14
General Target Operations . . . . . 12-15
Documentation . . . . . 12-15

# Introduction to Developer's Kit for Texas Instruments™ DSP

The Developer's Kit for Texas Instruments™ DSP 1.1 is a new product that is included on our product CD for the first time in Release 12.1

**Note** The Developer's Kit for Texas Instruments™ DSP 1.0 was released as a Web-downloadable product prior to Release 12.1. Version 1.1 of this product includes some important updates to Version 1.0, as summarized in "New Features" on page 12-4.

Composed of three component collections, the Developer's Kit lets you develop signal processing systems for the C6701 EVM, and use MATLAB to access and load data to and from the digital signal processor on a target signal processor. The three libraries are shown below.

| Product Name | Library Name | Description |
|---|---|---|
| Target for C6701 EVM | `c6701evmlib` | Contains the blockset needed to use Simulink and Real-Time Workshop to create applications for the C6701 EVM. Use this set of four blocks to configure the coder-decoder (codec) and LEDs on the C6701 EVM. Also lets you target Code Composer Studio to build new projects from Real-Time Workshop. |
| Link for Code Composer Studio IDE | `tiddk/@ccs/ccsdsp` | Contains the Code Composer Studio IDE (CCS IDE) interface software. Includes functions that enable you to use MATLAB to communicate with the CCS IDE. |
| Link for RTDX Interface | `tiddk/@ccs/rtdx` | Contains MATLAB objects and methods for communicating with target digital signal processors on boards installed in your PC, such as the C6701 EVM. With RTDX you can read and write data in real time to your target. |

The Developer's Kit lets you use Simulink to model floating-point digital signal processing algorithms from blocks in the DSP Blockset, and then use the Real-Time Workshop to build ANSI C code targeted to the TMS320C6701 EVM (C6701 EVM). The Developer's Kit takes the generated C code and uses Texas Instruments (TI) tools to build C6701 EVM-specific machine code. The build process downloads the targeted machine code to the evaluation module and runs the executable on the TMS320C6701 Digital Signal Processor.

After downloading the code to the board, your digital signal processing application or algorithm runs automatically on the evaluation module. You have the option of compiling your model in CCS IDE without downloading the code to the target.

Texas Instruments produces the C6701 EVM to help you create DSP applications for the Texas Instruments TMS320C6701 processor. You can create, test, and deploy your signal processing algorithms or filters on the target processor without the difficulties inherent in starting with the digital signal processor itself and building the support hardware to test the application on the processor. Instead, the C6701 EVM provides the input hardware, output hardware, timing circuitry, memory, and power for the C6701 digital signal processor. Texas Instruments provides the software tools, such as the C compiler, linker, and assembler, and Code Composer Studio (CCS), for you to use your PC to develop, download, and test your algorithms and applications on their C6701 EVM. You can launch Code Composer Studio Integrated Development Environment (CCS IDE) directly from the Developer's Kit. When you use the targeting option, the Developer's Kit launches the integrated development environment and populates a new development project with the files from your Real-Time Workshop build process. You select whether the process downloads your code to the target, or stops after it builds the new project.

# New Features

**Note** The Developer's Kit for Texas Instruments™ DSP 1.0 was released as a Web-downloadable product prior to Release 12.1. Version 1.1 of this product includes some important updates to Version 1.0, as summarized below.

## New Link Functions for CCS IDE

Version 1.1 of the Developer's Kit adds five functions that extend the program debugging and project management capabilities of the tool.

| Function | Description |
|---|---|
| `add` | Places a file in the current CCS project. Use this function to add the specified file in your current project in CCS. Supports adding C/C++ files, ASM files, object and library files, linker command files, and DSP/BIOS files. |
| `delete` | Removes debug points in CCS. You can remove breakpoints, probe points, and profile points. To identify the point to delete you specify either the address or line number. |
| `insert` | Adds debug points in CCS. You can add breakpoints, probe points, and profile points. To identify the location and type of point to insert you specify either the address or line number. |
| `isrtdxcapable` | Determines and returns whether the target associated with a CCSDSP object supports RTDX communications. |
| `remove` | Removes files from the current CCS project. With this function you can remove any type of file contained in the project. |

## New Interface Between the Developer's Kit and FDATool

If you own the Filter Design Toolbox or the Signal Processing Toolbox, you can design filters in Filter Design and Analysis Tool (FDATool) and export those filter designs directly to CCS IDE and to your target. You do not need to export the filters to your MATLAB workspace first.

With this link, you can design filters, export them to your target, and test them in your signal processing application. In addition, you can tune your filter performance in FDATool, and repeatedly export the modified filter to your target to test its new capability. Using FDATool with the Developer's Kit takes your filter design capabilities to a new level of flexibility and efficiency.

## Unified CCS IDE and C6701 EVM Targeting

In Version 1.1, the separate CCS target has been eliminated. To target CCS IDE is the same as targeting the C6701 EVM, without compiling, linking, or writing the program to the target. Removing the separate target results in cleaner processing using a common code path. As a result of unifying the targets, the **Simulation Parameters** dialog contains a new option on the **Real-Time Workshop** page.

Instead of selecting the CCS target file `ccs.tlc` for the **Target Configuration** to target CCS, you select Create_CCS_Project for the **Build Action** under the **Category:** TI C6701EVM runtime. When you click **Make Project**, the Developer's Kit uses Real-Time Workshop to create ANSI C code from your model and uses the code to populate a new project in CCS IDE.

---

**Note** Models that you built in Version 1.0 of the Developer's Kit use the `ccs.tlc` target TLC file. In Version 1.1, that file is no longer on the list of available targets in Real-Time Workshop. When you try to build models you developed in Version 1.0 in Version 1.1, you get an error that Real-Time Workshop cannot find the file `ccs.tlc`. To correct the error, browse for the `ti_c6701evm.tlc` target file to replace the missing file. Then rebuild your target.

---

## New Run-Time Libraries

To reduce the size of the generated .out files when you build models for targeting, and to improve the efficiency of the resulting file, the Developer's Kit now uses run-time libraries to include many block functions in the .out file. When your model uses one of a number of DSP blocks, the .out file generated by the compile uses the .obj file for the block from the run-time library that is appropriate for the target. In Version 1.0. the .out file included all versions of the block algorithm, rather than just the one required for the specified target. For example, now when you use the FFT block in a model targeted to a 16-bit processor, only the 16-bit FFT code appears in the .out file, instead of all 48 variants of FFT.

One side effect of this architectural change shows up in the Simulink simulation parameters. When you build a model with the default settings, compiling the model operates as before. If you change one or more of the parameters from the default setting, such as switching to little-endian from big-endian, the compile time can grow somewhat. Compiling a model with simulation parameter settings that are not the defaults can raise the compile time to as much as ten minutes or more for complex models.

## Architecture Change in CCSDSP Object

To more easily support future target development and to speed up the processing associated with ccsdsp objects (created when you use the function ccsdsp), the ccsdsp object does not include user-accessible ActiveX handles, such as dspuser. Removing the handles reduces the risks associated with using handles and the housekeeping required in managing your handles to ensure that handles do not cause problems if they are left unattended.

To use the handles as they existed in the earlier product release, for example to perform additional ActiveX/COM interactions with CCS IDE, you can regenerate the handles. The following MATLAB script recreates the handles for a specified board and processor.

```
% h = structure that will hold the CCS/RTDX handles
boardnum = 0   % Supplied desired board number
procnum = 0    % and processor number

% Create activeX handles to Code Composer
h.app = actxserver('CodeComposer.Application');
h.dspboards = get(h.app,'DspBoards');
```

```
h.dspboard = invoke(h.dspboards,'Item',boardnum);
h.dsptasks = get(h.dspboard,'DspTasks');
h.dsptask  = invoke(h.dsptasks,'Item',procnum);
invoke(h.dsptask,'OpenDebugControlWindow');
h.dspuser   = invoke(h.dsptask,'createdspuser');
```

## Updated Function writemsg Help Text

The new, faster code in MATLAB Release 12.1 exposes an explicit timeout feature that was not included in Release 12.0. Refer to the Help browser entry for `writemsg` for details.

## Major Bug Fixes

The following problems identified in Version 1.0 have been fixed in this release:

- Your C6701 EVM no longer has to be the first board in your computer. In Version 1.0, your evaluation module had to be board 0, with its processor as processor 0. In this release, you can install your C6701 EVM in any position, and identify it specifically when you use it as your target. Use the **Real-Time Workshop** page of the **Simulation Parameters** dialog to specify the target board.
- You do not need to close CCS IDE when you target your C6701 EVM.
- Resetting your board and closing CCS IDE are not required when you switch between targeting and linking or linking and targeting operations.

# Upgrading from an Earlier Release

Version 1.1 of the Toolbox incorporates significant changes to the targeting options, the CCSDSP object, and the way Simulink blocks are generated.

## New Libraries at Run-Time

The Developer's Kit now uses run-time libraries to include many block functions in the .out file. Refer to "New Run-Time Libraries" on page 12-6 for more information about how this change affects your existing models.

## CCSDSP Changes

To support future development, ccsdsp objects no longer use ActiveX handles as they did in Version 1.0. Refer to "Architecture Change in CCSDSP Object" on page 12-6 for further information about this change.

## Unified Targeting

With the Version 1.1 release, the separate CCS target is no longer available. Models developed with the 1.0 release generate an error when you try to build them in Version 1.1. For information about resolving the RTW error when you build, refer to "Unified CCS IDE and C6701 EVM Targeting" on page 12-5.

# Known Software and Documentation Problems

This section updates the Developer's Kit for Texas Instruments DSP 1.1 documentation, presenting the known problems with the software and documentation. Each entry in the following sections describes a situation you may encounter when you use the Developer's Kit.

## Operating Systems

### Windows 98

When you create a link to Texas Instruments Code Composer Studio (CCS) immediately after you reboot your Microsoft Windows 98 machine, it may take a very long time to create the link. To avoid the wait, launch CCS from your desktop before you create the link.

### Unsupported Platforms

The Developer's Kit, Version 1.1, is not supported on Microsoft Windows 2000 or Code Composer Studio 2.0.

## Linking

- Having CCS visible on your desktop during CCS read or write operations slows down the data transfer process. Generally, transfer rates fall by a factor of four when CCS IDE is visible. When you do not need to interact with CCS IDE, use visible to force CCS to the background to avoid the slowdown.

  ```
  cc = ccsdsp;
  cc.visible(0); % Pushes CCS IDE to the background.
  ```

  If your work requires that you interact with CCS IDE during linking operations, place CCS IDE in the background temporarily when your program transfers large amounts of data either direction. For example, the following code demonstrates changing the CCS IDE visibility state before and after large read and write operations.

```
cc = ccsdsp:
.
.         % Interact with CCS IDE.
.
.
vstate = cc.isvisible; % Store visibility state for CCS.
cc.visible(0);          % Force CCS to the background.
x = cc.read('FF00','int32',[20 20],20); % Large read transfer.
cc.write('FF00',int32(randn(20)),20); % Large write transfer.
cc.visible(vstate); % Return CCS to its previous visibility state.
.
.         % Continue to interact with CCS IDE.
.
.
```

- CCS limits memory read and write data transfers to 32 KB. This restriction does not affect RTDX.
- Before you use the links, run the GEL initialization that is appropriate for your target processor. For RTDX communications to work properly on the C5402 DSK, you must initialize the board. To initialize your DSK, select **C54x submenu --> C5402_DSK_Init** from the **GEL** option on the CCS menu bar.

  To automate this process as part of GEL reset, edit `C5000.gel` as follows:

  **a** In the `.gel` file, locate the section beginning `hotmenu C5402_DSK_Init`.

  **b** In the section, highlight and copy all instructions between the braces, `{ }`.

  **c** Paste your copied text into the startup function at the beginning of the `.gel` file.

  Subsequent GEL resets will run the DSK initialization.
- During memory transfer operations, you may receive a timeout error telling you that the time allowed for the operation expired. In most cases, the transfer has been successful in spite of the error message. Once the transfer operation starts, it cannot be stopped or changed. Time may expire if the transfer completes but the completion message is delayed. Try setting the timeout value to a large number of seconds to avoid the error message.

- When you configure your target code to perform RTDX write operations (write data from the target to the host) on the C5402 DSK, you must poll the 'successful emptying of the write buffer' status. The C6000, which allows polling or host-initiated interrupts, does not require polling.

  The following code provides an example of the polling syntax, where ochan is the write channel.

```
write(&ochan, &data, sizeof(data));
while ( RTDX_writing != NULL ) {
#if RTDX_POLLING_IMPLEMENTATION
    RTDX_Poll();
#endif
}
```

- On some occasions, CCS opens information dialogs in the background. You must respond to the dialogs before you can continue to work in MATLAB or CCS. Until you respond to the dialogs, MATLAB will not respond to input.

## Targeting

- For reasons related to the Texas Instruments C compiler, you cannot use three blocks in your Simulink model for targeting. They are:
  - Singular Value Decomposition block in the library dspfactors
  - Pseudoinverse block in the library dspinverses
  - SVD Solver block in the library dspsolvers

  These blocks are not included in the run-time libraries and do not build when you build your model.

- CCS cannot open some header files (files with the `.h` extension) when you use the Developer's Kit to build a project in CCS IDE. When you try to view the header files in CCS, you get an error "`filename.h was not found`," telling you that CCS could not find the files. Although CCS reports the error, the header files are being used in the compile and build process. To make CCS IDE aware of the files, perform the following steps to update the include file paths in your CCS project.

1 Locate the file *`modelname`*`.bld`, where *`modelname`* is the name of your Simulink model.

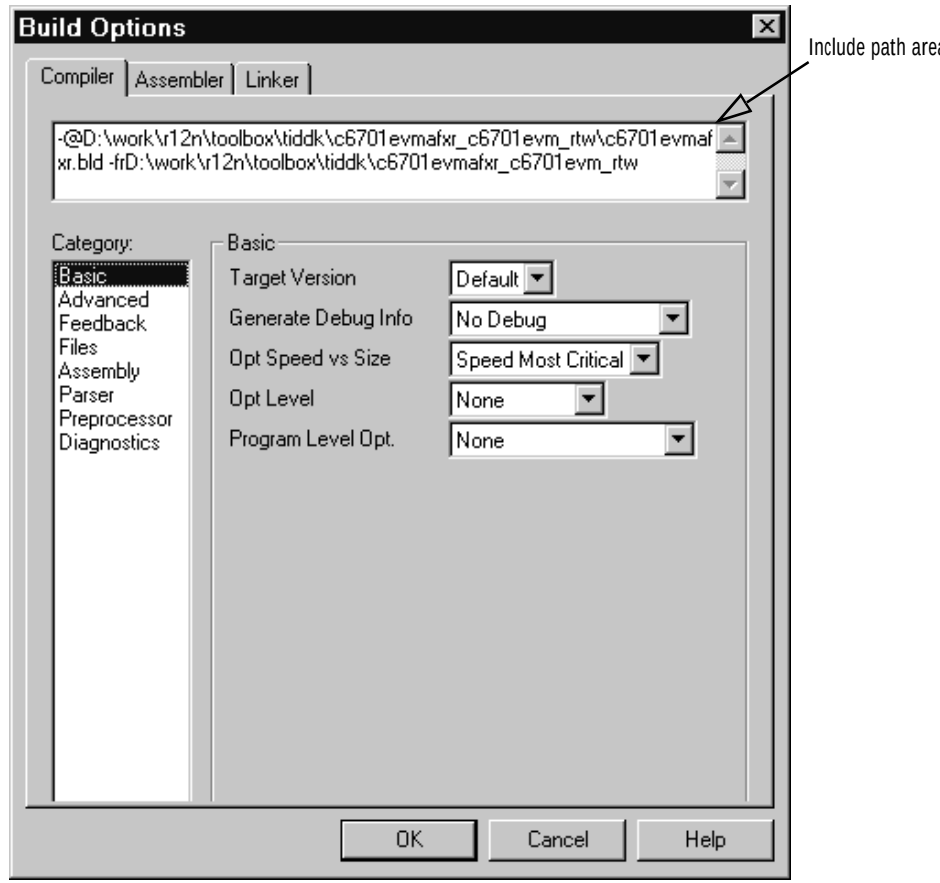
2 Open *modelname*.bld for editing in MATLAB.

3 Within *modelname*.bld, find the section /*add include paths*/.

4 Highlight and copy the paths listed in the section. You are going to use these paths to update the path definition for your project in CCS IDE.

5 Switch to CCS IDE.

6 Select **Project->Options** from the menubar. Click **Compiler** on the **Build Options** dialog to see the dialog shown here.



7 In the Include path area marked in the figure, append the text you copied from *modelname*.bld to the end of the existing path. Do not overwrite the existing path entries.

8 Click **OK** to close the **Build Options** dialog.

9 Select **Project->Scan All Dependencies...** from the CCS IDE menubar.

CCS IDE scans the new include paths and updates the file information. Now you can view all of the header files in the Include directory in CCS IDE.

---

**Note** After you update the search path, save the project as a CCS workspace with the .wks extension by selecting **File->Workspace->Save workspace...** from the CCS menubar if you plan to rebuild your model or project. If you do not save your project, when you build again with this model you lose the updated file path information.

---

## Demos

In MATLAB Demos, the link for the RTDX demonstration program named LMS Adaptive Filtering runs slowly on DSP Starter Kit (DSK) boards. Design features of the board cause the slow processing. Using the Texas Instruments XDS 510 JTAG/PC Controller, or an equivalent emulator, to connect your host to the DSK can alleviate the problem.

## Datatypes

C5000 and C6000 family processors handle integer datatypes differently. On C5000 family processors, 8-bit values do not have unique addresses. The C5000 family processors use 16-bit addresses natively. When you read and write to C5000 family processors, 8-bit integers in MATLAB are stored as 16-bit integers on the processor. In detail, the differences are:

- When you write an 8-bit integer from MATLAB to a C5000 family processor, MATLAB pads the value to 16 bits and the processor stores the value as a 16-bit integer.

- When you read an 8-bit integer from the processor into MATLAB, the read function takes 16 bits from processor memory, discards the eight most significant bits, and returns the remaining eight bits to MATLAB as the integer value.

For these reasons, scripts that run on C6000 family processors may not run without modification on C5000 family processors.

## General Target Operations

When your target digital signal processor is running, CCS uses up to 99% of the CPU cycles on your PC. Consequently, some other operations may be very slow, such as running the MATLAB help system or processing other applications.

## Documentation

The Help browser does not provide online documentation for the following functions:

- `add`
- `delete`
- `insert`
- `isrtdxcapable`
- `remove`

Access help information by entering

```
helpwin ccsdsp/functionname
```

at the MATLAB prompt. For example, for information about insert, type

```
helpwin ccsdsp/insert
```

# DSP Blockset 4.1 Release Notes

**New Features** . . . . . 13-2
New Digital Filter Design Block . . . . . 13-2
Major Individual Block Enhancements . . . . . 13-3
Blocks with Enhanced Data Type Support . . . . . 13-5
Blocks with Tunability Enhancements . . . . . 13-7
Enhancements to Blocks Implemented with S-Functions . . 13-7
Real-Time Workshop Generated C Code Enhancements . . . 13-8
Changes to Filtering Block Libraries . . . . . 13-9
New and Renamed DSP Blockset Directories . . . . . 13-10

**Major Bug Fixes** . . . . . 13-11
Real-Time Workshop C Code Generated from DSP Blockset 13-11
Nontunable Block Parameters . . . . . 13-11
Chirp Block . . . . . 13-11
Matrix Viewer Block . . . . . 13-11
Random Source Block . . . . . 13-12
Stack and Queue Blocks . . . . . 13-12
Window Function Block . . . . . 13-12

**Platform Limitations for HP and IBM** . . . . . 13-13

**Upgrading from an Earlier Release** . . . . . 13-14

**Known Software Problems** . . . . . 13-15
Real-Time Workshop C Code Generation Limitations . . . 13-15
Nonzero Simulation Start Times Not Supported . . . . . 13-15
Triggered and Enabled Subsystem Support . . . . . 13-16
Digital Filter Design Block Limitations . . . . . 13-16
Magnitude FFT Block Limitations . . . . . 13-17

# New Features

This section introduces the new features and enhancements added in the DSP Blockset 4.1 (Release 12.1) since DSP Blockset 4.0 (Release 12.0).

For information about the DSP Blockset features incorporated in recent releases, see the following documents:

- *Release Notes for Release 12*
- DSP Blockset `Readme.m` file — Open by typing `whatsnew dspblks` or `info dspblks` at the MATLAB command line.

The DSP Blockset 4.1 new features are discussed in detail in the following sections:


- "New Digital Filter Design Block" on page 13-2
- "Major Individual Block Enhancements" on page 13-3
- "Blocks with Enhanced Data Type Support" on page 13-5
- "Blocks with Tunability Enhancements" on page 13-7
- "Enhancements to Blocks Implemented with S-Functions" on page 13-7
- "Real-Time Workshop Generated C Code Enhancements" on page 13-8
- "Changes to Filtering Block Libraries" on page 13-9
- "New and Renamed DSP Blockset Directories" on page 13-10


## New Digital Filter Design Block

The new Digital Filter Design block integrates the Filter Design and Analysis Tool graphical user interface (FDATool) with the DSP Blockset. Double-clicking the block opens FDATool, which you can use to quickly design, import, or directly specify a digital FIR or IIR filter. Using the specified filter, the block filters the input signal and outputs the result.

Currently, the Digital Filter Design block only supports direct-form II transposed filter structures, so the block converts the structures of all imported filters to direct-form II transposed. In addition, the block does not support the **Set Quantization Parameters** tab, which is usually available in FDATool when you install the Filter Design Toolbox.

The Digital Filter Design block replaces many of the blocks from DSP Blockset 4.0 (Release 12). For details, see “Changes to Filtering Block Libraries” on page 13-9.

## Major Individual Block Enhancements

The following sections summarize the major individual block enhancements for DSP Blockset 4.1:

- “FFT Block”
- “IFFT Block”
- “Magnitude FFT Block”
- “Overlap-Add FFT Filter and Overlap-Save FFT Filter Blocks”
- “Random Source Block”
- “Sine Wave Block”
- “Unwrap Block”
- “Window Function Block”

### FFT Block

- You can now set the block to compute twiddle factors by table lookup or by calling trigonometric math library functions. In table lookup mode, the block precomputes and stores a table that you can optimize for memory or speed. For an N-point FFT, a table optimized for speed yields a length-3N/4 table, and a table optimized for memory yields a length-N/4 table.
- A new option lets you set the block to output in bit-reversed or linear order. Outputting in bit-reversed order reduces computation time by eliminating the output data descrambling step.
- The block processes real-valued inputs using more efficient algorithms.
- The block generates smaller Real-Time Workshop C code with improved run-time efficiency through use of in-place algorithms.
- The block now supports single-precision (32-bit) floating-point input and output signals (in table lookup mode) for both simulation and Real-Time Workshop C code generation.

For more information, see the FFT block reference page.

### IFFT Block

- You can now set the block's twiddle factor computation method as in the FFT block (see the earlier FFT block enhancement description).
- A new option lets you provide inputs in both linear and bit-reversed order. Using inputs in bit-reversed order reduces block computation time by eliminating the initial input data scrambling step.
- You can now explicitly specify conjugate-symmetric input signals to significantly increase processing efficiency and to get a real-valued output signal.
- You can now skip normalization by the IFFT length, N, when you do not need to output a normalized signal, increasing processing efficiency.
- The block generates smaller Real-Time Workshop C code with improved run-time efficiency through use of in-place algorithms.
- The block now supports single-precision (32-bit) floating-point input and output signals (in table lookup mode) for both simulation and Real-Time Workshop C code generation.

For more information, see the IFFT block reference page.

### Magnitude FFT Block

- You can now set the block to compute the magnitude FFT or magnitude squared FFT
- The block uses the new, more efficient FFT block algorithms.

For more information, see the Magnitude FFT block reference page.

### Overlap-Add FFT Filter and Overlap-Save FFT Filter Blocks

By using the enhanced FFT and IFFT blocks, the Overlap-Add FFT Filter and Overlap-Save FFT Filter blocks are more efficient. (Both blocks now skip data descrambling and scrambling operations by processing data in bit-reversed order, and use specialized, more efficient algorithms for real-valued input signals.)

### Random Source Block

- You can now set the repeatability of the block output to **Not repeatable**, **Repeatable**, or **Specify seed**. The default mode is **Not repeatable**, so that when you add new Random Source blocks from the block library to a model, the blocks automatically generate different outputs (without requiring you to specify a seed).
- The block now uses Simulink run-time parameters, greatly enhancing its Real-Time Workshop generated C code (see "Enhancements to Blocks Implemented with S-Functions" on page 13-7).

### Sine Wave Block

You can now optimize the Sine Wave block's table lookup computation method for speed or memory. For speed optimization, the block uses a full-length table, while for memory optimization, it uses a 1/4-length table.

### Unwrap Block

You can now set the Unwrap block to phase unwrap across successive inputs for both frame-based and sample-based input signals.

### Window Function Block

- The Window Function block now generates smaller, more efficient Real-Time Workshop C code by using run-time parameters (see "Enhancements to Blocks Implemented with S-Functions" on page 13-7).
- The block now supports single-precision (32-bit) floating-point input and output signals.

## Blocks with Enhanced Data Type Support

All DSP Blockset 4.1 blocks support the double data type in both simulation and in Real-Time Workshop C code generation. Many blocks now support all Simulink built-in data types, as well as the fixed-point data type and properly defined Simulink custom data types in both simulation and in Real-Time Workshop C code generation. To see which data types a particular block supports, check the "Supported Data Types" section of its reference page.

The data signal input and output ports (noncontrol input and output ports) of the following blocks now support all built-in, fixed-point, and other properly defined Simulink custom data types:

- Buffer
- Check Signal Attributes
- Contiguous Copy
- Convert 1-D to 2-D
- Convert 2-D to 1-D
- Delay Line
- Display
- Downsample
- Extract Diagonal
- Extract Triangular Matrix
- Flip
- Frame Status Conversion
- Inherit Complexity
- Integer Delay
- Matrix Viewer
- Multiport Selector
- Pad
- Permute Matrix (also, `index` input port accepts signed and unsigned integer and double- and single-precision floating-point)
- Repeat
- Sample and Hold
- Signal To Workspace
- Spectrum Scope
- Submatrix
- Time Scope
- Toeplitz
- Transpose
- Triggered Delay line
- Triggered To Workspace
- Unbuffer
- Upsample
- Variable Integer Delay
- Variable Selector (also, `index` input port accepts signed and unsigned integer and double- and single-precision floating-point)
- Vector Scope
- Zero Pad

The data signal input and output ports (noncontrol input and output ports) of the following blocks now support all Simulink built-in data types:

- Constant Diagonal Matrix
- DSP Constant
- Signal From Workspace
- Triggered Signal From Workspace

The data signal input and output ports (noncontrol input and output ports) of the following blocks now support the single-precision (32-bit) floating-point data type:

- dB Gain
- Discrete Impulse
- FFT (in table lookup mode)
- IFFT (in table lookup mode)
- Matrix Multiply
- Matrix Sum
- Maximum
- Polynomial Stability Test
- Quantizer
- Uniform Decoder
- Uniform Encoder
- Window Function

## Blocks with Tunability Enhancements

The following blocks have enhanced tunability:

- Chirp — **Target time** parameter is now tunable.
- Counter — **Maximum count** parameter is now tunable.
- Mean — **Reset port** parameter is now tunable.
- Pad — **Value** parameter is now tunable.

## Enhancements to Blocks Implemented with S-Functions

Many of the DSP Blockset blocks are S-function blocks (implemented with Simulink S-functions). The following sections summarize some of the major DSP Blockset 4.1 enhancements that improve S-function block performance and help you to write better S-functions.

### Use of Simulink Run-Time Parameters

Many S-function blocks now generate better optimized Real-Time Workshop C code by using Simulink run-time parameters (RTPs) for both mask dialog and internal parameters. Blocks that use RTPs include those with filter coefficient parameters and initial condition parameters.

### Better Memory Management for Simulation and Code Generation

Memory management enhancements for S-function block simulation and code generation now provide simulation code requiring less memory, and smaller Real-Time Workshop generated C code that is more efficient at run-time. Memory management enhancements include the use of contiguous input and output arrays, reusable inputs, overwritable arrays, and in-place algorithms.

### New S-Function Simulation Support Directories

Various extensible C-callable library functions, macros, and definitions are now included in the following directories:

- `toolbox/dspblks/include` — header files (prototypes and definitions)
- `toolbox/dspblks/src/sim` — simulation library source code
- `toolbox/dspblks/lib` — prebuilt library (`.lib`) files for certain combinations of platforms and compilers only

## Real-Time Workshop Generated C Code Enhancements

DSP Blockset 4.1 enhancements that provide better Real-Time Workshop generated C code include the following:

- DSP Blockset can now build and link C-callable run-time libraries, providing for smaller, more efficient Real-Time Workshop generated C code.
- The use of run-time parameters in blocks implemented with S-functions (see “Enhancements to Blocks Implemented with S-Functions” on page 13-7) provides for the following Real-Time Workshop generated C code enhancements for blocks implemented using S-functions:
  - Parameter values are organized in a common structure.
  - Parameter values are located in a common header file (for example, `<model>_prm.h`).
  - Redundant copies of parameter values are reused or “pooled” in `<model>_prm.h` (when there are multiple instances of a block with the same run-time parameter values).

## Changes to Filtering Block Libraries

This section describes the changes to the contents and names of the DSP Blockset filtering block libraries.

### Replaced Filtering Block Libraries

The new **Filter Designs** block library in DSP Blockset 4.1 replaces the old **Filter Structures** and **Filter Designs** block libraries in DSP Blockset 4.0, and contains the following blocks:

- Analog Filter Design
- Biquadratic Filter
- Digital Filter Design
- Direct-Form II Transpose Filter
- Filter Realization Wizard
- Overlap-Add FFT Filter
- Overlap-Save FFT Filter
- Time-Varying Lattice Filter
- Time-Varying Direct-Form II Transpose Filter

### Replaced Filtering Blocks

The new Digital Filter Design block in the **Filter Designs** block library (see "New Digital Filter Design Block" on page 13-2) replaces the following DSP Blockset 4.0 blocks:

- Digital FIR Filter Design
- Digital FIR Raised Cosine Filter Design
- Digital IIR Filter Design
- Least Squares FIR Filter Design
- Remez FIR Filter Design

You can still access these blocks by typing `dspddes3` at the MATLAB command line, but they are no longer in the DSP Blockset Filter Designs block library.

## New and Renamed DSP Blockset Directories

New and renamed DSP Blockset 4.1 product directories and their contents are summarized in the following table. (The directories in the table are all in the `toolbox/dspblks` directory.)

| Directory Name | Directory Contents |
|---|---|
| `include` | DSP Blockset simulation and run-time support function prototypes, macros, definitions, etc. |
| `lib` | Prebuilt DSP Blockset simulation and run-time support libraries |
| `src/rt` | DSP Blockset run-time support library source code |
| `src/sim` | DSP Blockset S-function simulation support library source code |
| `src/v2` | Selected DSP Blockset Version 1.x and Version 2.x S-function source code |
| `src/v3` | Selected DSP Blockset Version 3.x S-function source code |
| `src/v4` | Selected DSP Blockset Version 4.x S-function source code |

# Major Bug Fixes

The following sections describe some of the major DSP Blockset 4.1 bug fixes:

- "Real-Time Workshop C Code Generated from DSP Blockset"
- "Nontunable Block Parameters"
- "Chirp Block"
- "Matrix Viewer Block"
- "Random Source Block"
- "Stack and Queue Blocks"
- "Window Function Block"

## Real-Time Workshop C Code Generated from DSP Blockset

- Real-Time Workshop C code generated from the blockset now uses the `sizeof` function rather than hard-coding word sizes for nonstandard code generation targets (such as those that do not treat characters as 8-bit bytes).
- The generated C code for most blocks no longer generates compiler warnings for most supported targets.

## Nontunable Block Parameters

All nontunable block parameters are now uneditable in the Simulink mask UI during simulations.

## Chirp Block

The Chirp block now has correct behavior in bidirectional sweep mode, and correctly interprets target frequency and sweep time parameters.

## Matrix Viewer Block

The Matrix Viewer block now displays the color bar.

## Random Source Block

The Random Source block seed initialization better supports multiple copies of the block in one Simulink model.

Also, the block no longer sets the seed of the MATLAB random number generator function during block initialization.

## Stack and Queue Blocks

The Stack and Queue blocks now correctly handle simultaneously triggered control ports (such as the Push and Pop ports).

## Window Function Block

The Window Function block now errors out for non-floating-point inputs to avoid possible segmentation violations.

# Platform Limitations for HP and IBM

On the HP and IBM platforms, the DSP Blockset has the same simulation limitations as Simulink, and the same code generation limitations as Real-Time Workshop.

## Upgrading from an Earlier Release

There are no issues for upgrading from DSP Blockset 4.0 (Release 12) to DSP Blockset 4.1 (Release 12.1).

For information about upgrading from an earlier version than DSP Blockset 4.0, see the *Release Notes for Release 12*.

# Known Software Problems

The following sections describe some of the known DSP Blockset problems and limitations:

- "Real-Time Workshop C Code Generation Limitations"
- "Nonzero Simulation Start Times Not Supported"
- "Triggered and Enabled Subsystem Support"
- "Digital Filter Design Block Limitations"
- "Magnitude FFT Block Limitations"

## Real-Time Workshop C Code Generation Limitations

C code generated from DSP Blockset blocks using Real-Time Workshop has the following limitations:

- DSP Blockset generally does not support the **Loop rolling threshold** parameter (visible when the **Category** parameter is set to **General code generation options**) in the **Real-Time Workshop** tab of the Simulink **Simulation Parameter**s dialog.
- Currently, the DSP Blockset run-time library `dsp_rt.lib` does not always rebuild when you change compiler or linker options. Thus, before changing compiler and linker options, you should delete `dsp_rt.lib` and all associated object files. You can change compiler and linker options in the Simulink **Simulation Parameters** dialog's **Real-Time Workshop** tab by setting `make` command arguments such as `OPTS` or `OPT_OPTS` in the **Make command** parameter (for example, `make_rtw OPTS="-DMYDEFINE=1"`).

## Nonzero Simulation Start Times Not Supported

The DSP Blockset does not support nonzero simulation start times. When running a simulation with DSP Blockset blocks, you must set the **Start time** option in the Simulink **Simulation Parameters** dialog's **Solver** tab to `0`.

## Triggered and Enabled Subsystem Support

Triggered or enabled subsystems currently do not support the following blocks:

- Buffer
- Counter
- Downsample
- Dyadic Analysis Filter Bank
- Dyadic Synthesis Filter Bank
- FIR Decimation
- FIR Interpolation
- Histogram
- Kalman Adaptive Filter
- LMS Adaptive Filter
- LU Solver
- Maximum
- Mean
- Minimum
- N-Sample Enable
- N-Sample Switch
- Overlap-Add FFT Filter
- Overlap-Save FFT Filter
- Permute Matrix
- Queue
- Repeat
- RLS Adaptive Filter
- RMS
- Stack
- Standard Deviation
- Time-Varying Direct-Form II Transpose Filter
- Time-Varying Lattice Filter
- Unbuffer
- Upsample
- Variable Fractional Delay
- Variable Integer Delay
- Variable Selector
- Variance
- Wavelet Analysis
- Wavelet Synthesis

## Digital Filter Design Block Limitations

The Digital Filter Design block has the following limitations:

- The Filter Design and Analysis Tool graphical user interface (FDATool), opened from the Digital Filter Design block, currently only supports direct-form II transposed filter structures. Thus, the Digital Filter Design block's FDATool converts the structures of all imported filters to direct-form II transposed.

- When you install the Filter Design Toolbox, FDATool, opened from the MATLAB command line, gets a new tab, **Set Quantization Parameters**. However, the Digital Filter Design block's FDATool does not currently support the **Set Quantization Parameters** tab.
- The Digital Filter Design block currently allows you to edit nontunable FDATool filter specification parameters. When you edit nontunable parameters during a simulation, the filter still updates in FDATool to reflect your changes. However, during simulation, FDATool only updates the block icon (which displays the frequency response of the updated filter). FDATool sends the new filter coefficients to the block only after the simulation stops running.

For details about the Digital Filter Design block, see the previous section, "New Digital Filter Design Block" on page 13-2.

## Magnitude FFT Block Limitations

The Magnitude FFT block currently does not support 2-D sample-based 1-by-N row vector inputs.

14

# Excel Link 1.1.3 Release Notes

**New Features** . . . . . 14-2
License Manager Control . . . . . 14-2
Additional Error Detection . . . . . 14-2

**Known Software Problems** . . . . . 14-3
Enabling the MATLAB Desktop . . . . . 14-3

# New Features

This section introduces the new features and enhancements added to Excel Link 1.1.3.

## License Manager Control

As with other MATLAB-related products, Excel Link installation is now under control of the FLEXlm license manager. Implementation of license manager control has necessitated a change in the directory location of several files created during the installation process.

---

**Note** The notation `<matlab>` represents the MATLAB root directory, the directory where MATLAB is installed on your system.

---

| File | Purpose | Previous Location | New Location |
|---|---|---|---|
| `excllink.xla` | Excel Link add-in | `<matlab>\exlink` | `<matlab>/toolbox/exlink` |
| `ExliSamp.xls` | Excel Link samples | `<matlab>\exlink` | `<matlab>/toolbox/exlink` |

## Additional Error Detection

Excel Link now produces a cell error message if your matrix exceeds the Excel limit of 256 columns or 65536 rows.

# Known Software Problems

## Enabling the MATLAB Desktop

Excel Link uses MATLAB as an ActiveX automation server. Consequently, when started from Excel Link, MATLAB comes up minimized with the MATLAB desktop disabled.

To enable the MATLAB desktop, you can execute the MATLAB command `desktop` directly in the command window or execute it indirectly using the Excel Link function `MLEvalString`.

To enable the MATLAB desktop from Excel, type the following in any cell.

```
=mlevalstring("desktop")
```

15

# Filter Design Toolbox 2.1 Release Notes


**New Features** . . . . . 15-2
New Adaptive Filtering Functions . . . . . 15-2
New FIR Filter Design Functions . . . . . 15-3
New Filter Transformation Functions . . . . . 15-3
Transformations Option in FDATool . . . . . 15-4
New Analysis Method . . . . . 15-4
New Context-Sensitive Help for Quantization . . . . . 15-5

**Platform Limitations for HP 10.2** . . . . . 15-6

**Upgrading from an Earlier Release** . . . . . 15-7
Obsolete Functions in Version 2.0 . . . . . 15-7

**Known Software and Documentation Problems** . . . . 15-8
Switching Between Design and Quantization Modes in FDATool . . . . . 15-8
Help for Filter Transformations in FDATool . . . . . 15-8

# New Features

This section introduces the new features and enhancements added in the Filter Design Toolbox 2.1 since Filter Design Toolbox 2.0 (Release 12).

For information about Filter Design Toolbox features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## New Adaptive Filtering Functions

The Filter Design Toolbox 2.1 includes these new filter design functions:

- `adaptkalman` — Use a Kalman filtering algorithm in an adaptive filter role
- `adaptlms` – Use an LMS-based adaptive algorithm in an adaptive filter role
- `adaptnlms` – Use a normalized LMS-based adaptive algorithm in an adaptive filter role
- `adaptrls` – Use an RLS-based adaptive algorithm in an adaptive filter role
- `adaptsd` – Use the sign-data variant of the LMS-based adaptive algorithm in an adaptive filter role
- `adaptse` – Use the sign-error variant of the LMS-based adaptive algorithm in an adaptive filter role
- `adaptss` — Use the sign-sign variant of the LMS-based adaptive algorithm in an adaptive filter role

To support you when you use these new functions, the Toolbox includes corresponding functions for initializing the associated adaptive filter algorithm.

- `initkalman` — Initialize the input argument `s` for the Kalman-based algorithm adaptive filter function
- `initlms` — Initialize the input argument `s` for the LMS-based adaptive filter function
- `initnlms` — Initialize the input argument `s` for the normalized LMS-based adaptive filter function
- `initrls` — Initialize the input argument `s` for the RLS-based adaptive filter function
- `initsd` — Initialize the input argument `s` for the sign-data variant of the LMS-based adaptive filter function

- `initse` — Prepare the input arguments for the sign-error variant of the LMS-based adaptive filter function
- `initss` — Prepare the input arguments for the sign-sign variant of the LMS-based adaptive filter function

## New FIR Filter Design Functions

In addition to the new adaptive filtering capabilities, this version of the toolbox has four new filter design functions.

- `firhalfband` — Design *L*th-band filters where *L*=2. About half of the filter coefficients are zero so the filters are very efficient to calculate
- `firminphase` — Calculate the minimum-phase FIR spectral factor of a linear-phase FIR filter
- `firnyquist` — Design lowpass filters with certain specified coefficients in the transfer functions set to zero by design
- `ifir` — Design an interpolated FIR filter

FDATool includes `Halfband` and `Nyquist` entries under **Filter Type** so you can design and analyze these kinds of filters within FDATool.

## New Filter Transformation Functions

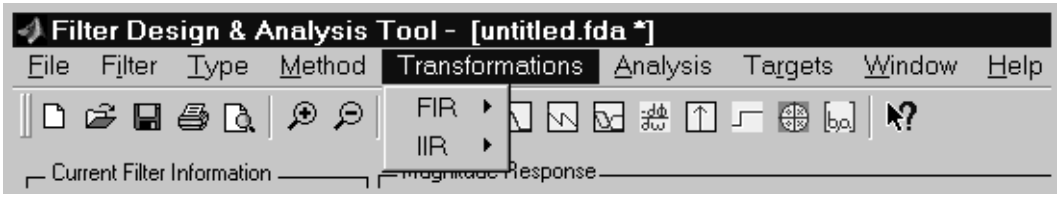
The Filter Design Toolbox 2.1 includes several new filter transformation functions. Each transformation maintains the ripple characteristics and passband/stopband shapes of the source filter while transforming to the new specification.

- `firlp2lp` — Transforms lowpass FIR filters to lowpass FIR filters with different passband width or cutoff specifications
- `firlp2hp` — Transforms lowpass FIR filters to highpass FIR filters
- `iirlp2bp` — Transforms lowpass IIR filters to bandpass IIR filters
- `iirlp2bs` — Transforms lowpass IIR filters to bandstop IIR filters
- `iirlp2hp` — Transforms lowpass IIR filters to highpass IIR filters
- `iirlp2lp` — Transforms lowpass IIR filters to lowpass IIR filters

The new **Transformations** option (refer to the next entry in this section) in FDATool provides access to these transformations from the GUI.

## Transformations Option in FDATool

When you use Filter Design and Analysis Tool (FDATool) with Filter Design Toolbox 2.1, the Toolbox adds a new menu to the FDATool menu bar. Named **Transformations**, the new selection provides direct access to the new FIR and IIR filter transformation functions added to the Toolbox in this release (refer to "New Filter Transformation Functions"). **Transformations** appears as shown in this figure only when you have installed the Filter Design Toolbox.




With this new option, FIR and IIR filters whose characteristics are particularly valuable become the sources for other FIR and IIR filters that share the same desirable features in a different passband configuration. Transforming a filter saves you from repeating the filter design process. For more information, refer to Transforming Filters in your *Filter Design Toolbox User's Guide*.

In the figure, the **Targets** option is available only when you have the Developer's Kit for Texas Instruments DSP installed.

## New Analysis Method

Filter Design Toolbox 2.1 includes one new analysis method — `zerophase` — that returns the zero-phase response of a linear-phase FIR filter.

`[hr,w] = zerophase(b)`, and the other available syntax options for `zerophase`, returns the zero-phase response `hr` and the frequency vector `w` (in rad/sample) at which `hr` is computed, given a linear-phase FIR filter `b`. The zero-phase response is evaluated at 512 equally spaced points on the upper half of the unit circle.

Note that the zero-phase response is always real, but it is not equivalent to the magnitude response. In particular, the zero-phase response can be negative; the magnitude response is always positive.

## New Context-Sensitive Help for Quantization

FDATool Quantization mode now includes context sensitive or What's This? help. For each option on the **Set Quantization Parameters** page, right-click to use the What's This? option to open a brief text description of the option.

## Platform Limitations for HP 10.2

Filter Design Toolbox 2.1 is not supported on the Hewlett-Packard 10.2 (HP 700) platform.

---

**Note** This is the same platform limitation that existed in the Filter Design Toolbox 2.0 (Release 12.0).

---

# Upgrading from an Earlier Release

There are no upgrade issues moving from the Filter Design Toolbox 2.1 from the Filter Design Toolbox 2.0.

---

**Note** The following upgrade information is the same as appeared in the *Release Notes for Release 12*.

---

## Obsolete Functions in Version 2.0

Filter Design Toolbox 2.0 makes obsolete the following functions that were part of Quantized Filter Design Toolbox:.

| Obsolete Function | Suggested Replacement |
|---|---|
| `propinfo` | Use `help constructor/propertyname` to get help about a function. Or use the Help browser. |
| `qfiltlog` | Use `qreport` to get information about quantized filters, quantized FFTs, and quantizers. |
| `qhelp` | Use `help constructor/propertyname` to get help about a function. Or use the Help browser. |

# Known Software and Documentation Problems

This section updates the Filter Design Toolbox 2.1 documentation, reflecting known Filter Design Toolbox 2.1 software or documentation problems.

## Switching Between Design and Quantization Modes in FDATool

After you scale a quantized filter, or convert a quantized filter to second-order sections, do not switch between quantized mode and filter design mode in FDATool. Selecting and clearing the **Turn quantization on** option (to shift between the quantization and design modes) after you have scaled or converted a quantized filter can corrupt the coefficients for your reference filter and your quantized filter.

To recover your reference filter design and quantized filter design if they have been corrupted, either:

- Import the filter from your workspace again
- Use FDATool to repeat the filter design

## Help for Filter Transformations in FDATool

Option **Transformations** in FDATool does not have What's This? help available. Use `helpwin` *`functionname`* for information about the transformation functions or use the Help browser to search the online documentation.

16

# Financial Time Series Toolbox 1.1 Release Notes

**New Features** . . . . . 16-2
New Functions . . . . . 16-2
Financial Time Series GUI . . . . . 16-2

**Major Changes to Functions** . . . . . 16-3

# New Features

**Note** The Financial Time Series Toolbox 1.1 was released in Web-downloadable format prior to Release 12.1. Version 1.1 of this product includes some important updates to Version 1.0, as summarized below. The Financial Time Series Toolbox 1.1 distributed on the Release 12.1 CD is the same as the version of the product distributed via the Web prior to Release 12.1.

## New Functions

| Function | Purpose |
|---|---|
| `bar` | Plot bar chart |
| `barh` | Plot horizontal bar chart |
| `bar3` | Plot 3-D bar chart |
| `bar3h` | Plot horizontal 3-D bar chart |
| `ftsgui` | Display financial time series graphical user interface main window |
| `ftsinfo` | Display information about a financial time series object |
| `log2` | Base 2 logarithm |

## Financial Time Series GUI

The new Financial Time Series GUI provides complete access to the full functionality of the Financial Time Series Toolbox.

# Major Changes to Functions

Candle plots are now correctly filled or open depending upon price movements.

The **Combine Axes** and **Zoom** operations of `chartfts` have been enhanced.

17

# Fixed-Point Blockset 3.1 Release Notes

**New Features** . . . . . 17-2
New Libraries . . . . . 17-2
New Blocks . . . . . 17-3
Renamed Blocks . . . . . 17-8
New Functions . . . . . 17-9
Data Type Support . . . . . 17-10
Changes to Functionality . . . . . 17-10

**Known Documentation Problem** . . . . . 17-11
Error in Printed Version of Documentation . . . . . 17-11

# New Features

This section introduces the new features and enhancements added in the Fixed-Point Blockset 3.1 (Release 12.1) since the Fixed-Point Blockset 3.0 (Release 12).

For information about Fixed-Point Blockset features that are incorporated from recent releases, see the *Release Notes for Release 12*.

This section is organized into these subsections


- "New Libraries" on page 17-2
- "New Blocks" on page 17-3
- "Renamed Blocks" on page 17-8
- "New Functions" on page 17-9
- "Data Type Support" on page 17-10
- "Changes to Functionality" on page 17-10


## New Libraries

The blocks in the Fixed-Point Blockset are now organized into twelve libraries:

- Bits
- Calculus
- Data Type
- Delays & Holds
- Edge Detect
- Filters
- Logic & Comparison
- LookUp
- Math
- Nonlinear
- Select
- Sources

To open the main Fixed-Point Blockset library, which contains the libraries listed above, type `fixpt_lib_3p1` at the MATLAB prompt.

## New Blocks

The Fixed-Point Blockset 3.1 includes the new blocks listed below.

| Block Name | Library | Description |
|---|---|---|
| Accumulator | Calculus | Compute a cumulative sum. |
| Accumulator Resettable | Calculus | Compute a cumulative sum with external Boolean reset. |
| Accumulator Resettable Limited | Calculus | Compute a limited cumulative sum with external Boolean reset. |
| Bit Clear | Bits | Set the specified bit of the stored integer to zero. |
| Bit Set | Bits | Set the specified bit of the stored integer to one. |
| Compare To Constant | Logic & Comparison | Determine if a signal is equal to the specified constant. |
| Compare To Zero | Logic & Comparison | Determine if a signal is equal to zero. |
| Cosine | LookUp | Implement a sine wave in fixed-point using a lookup table approach that exploits quarter wave symmetry. |
| Counter Free | Sources | Count up and overflow back to zero after the maximum value possible is reached for the specified number of bits. |
| Counter Limited | Sources | Count up, and wrap back to zero after outputting the specified upper limit. |
| Data Type Duplicate | Data Type | Set all inputs to the same data type. |
| Dead Zone Dynamic | Nonlinear | Set the input within the bounds to zero. |
| Decrement Real World | Math | Decrease the real world value of the signal by one |

| Block Name | Library | Description |
|---|---|---|
| Decrement Stored Integer | Math | Decrease the stored value of a signal by one |
| Decrement Time To Zero | Math | Decrease the real world value of the signal by the sample time, but only to zero. |
| Decrement To Zero | Math | Decrease the real world value of a signal by one, but only to zero. |
| Derivative | Calculus | Compute a discrete time derivative. |
| Detect Change | Edge Detect | Detect a change in a signal's value. |
| Detect Decrease | Edge Detect | Detect a decrease in a signal's value. |
| Detect Fall Negative | Edge Detect | Detect a falling edge when the signal's value decreases to a strictly negative value, and its previous value was nonnegative. |
| Detect Fall Nonpositive | Edge Detect | Detect a falling edge when the signal's value decreases to a nonpositive value, and its previous value was strictly positive. |
| Detect Increase | Edge Detect | Detect an increase in a signal's value. |
| Detect Rise Nonnegative | Edge Detect | Detect a rising edge when a signal's value increases to a nonnegative value, and its previous value was strictly negative. |
| Detect Rise Positive | Edge Detect | Detect a rising edge when a signal's value increases to a strictly positive value, and its previous value was nonpositive. |
| Difference | Calculus | Calculate the change in a signal over one time step. |
| Filter Direct Form I | Filters | Implement a Direct Form I realization of a filter. |
| Filter Direct Form I Time Varying | Filters | Implement a time varying Direct Form I realization of a filter. |

| Block Name | Library | Description |
|---|---|---|
| Filter Direct Form II | Filters | Implement a Direct Form II realization of a filter. |
| Filter Direct Form II Time Varying | Filters | Implement a time varying Direct Form II realization of a filter. |
| Filter First Order | Filters | Implement a discrete-time first order filter. |
| Filter Lead or Lag | Filters | Implement a discrete-time lead or lag filter. |
| Filter Real Zero | Filters | Implement a discrete time filter that has a real zero and no pole. |
| Increment Real World | Math | Increase the real world value of the signal by one. |
| Increment Stored Integer | Math | Increase the stored integer value of a signal by one. |
| Integrator Backward | Calculus | Perform discrete-time integration of a signal using the backward method. |
| Integrator Backward Resettable | Calculus | Perform discrete-time integration of a signal using the backward method, with external Boolean reset. |
| Integrator Backward Resettable Limited | Calculus | Perform discrete-time limited integration of a signal using the backward method, with external Boolean reset. |
| Integrator Forward | Calculus | Perform discrete-time integration of a signal using the forward method. |
| Integrator Forward Resettable | Calculus | Perform discrete-time integration of a signal using the forward method, with external Boolean reset. |
| Integrator Forward Resettable Limited | Calculus | Perform discrete-time limited integration of a signal using the forward method, with external Boolean reset. |

| Block Name | Library | Description |
|---|---|---|
| Integrator Trapezoidal | Calculus | Perform discrete-time integration of a signal using the trapezoidal method. |
| Integrator Trapezoidal Resettable | Calculus | Perform discrete-time integration of a signal using the trapezoidal method, with external Boolean reset. |
| Integrator Trapezoidal Resettable Limited | Calculus | Perform discrete-time limited integration of a signal using the trapezoidal method, with external Boolean reset. |
| Interval Test | Logic & Comparison | Determine if a signal is in a specified interval. |
| Interval Test Dynamic | Logic & Comparison | Determine if a signal is in a specified interval. |
| MinMax Running Resettable | Math | Determine the minimum or maximum of a signal over time. |
| Rate Limiter | Nonlinear | Limit the rising and falling rates of the signal. |
| Rate Limiter Dynamic | Nonlinear | Limit the rising and falling rates of the signal. |
| Repeating Sequence Interpolated | Sources | Output a discrete-time sequence and repeat, interpolating between data points |
| Repeating Sequence Stair | Sources | Output a discrete time sequence and repeat. |
| Sample Time Multiply | Calculus | Support calculations involving sample time. |
| Saturation Dynamic | Nonlinear | Bound the range of the input. |
| Scaling Strip | Data Type | Remove scaling and map to a built in integer. |
| Sine | LookUp | Implement a sine wave in fixed-point using a lookup table approach that exploits quarter wave symmetry. |
| State-Space | Filters | Implement discrete-time state space. |

| Block Name | Library | Description |
|---|---|---|
| Unit Delay | Delays & Holds | Delay a signal one sample period. |
| Unit Delay Enabled | Delays & Holds | Delay a signal one sample period. |
| Unit Delay Enabled External IC | Delays & Holds | Delay a signal one sample period. |
| Unit Delay Enabled Resettable | Delays & Holds | Delay a signal one sample period. |
| Unit Delay Enabled Resettable External IC | Delays & Holds | Delay a signal one sample period. |
| Unit Delay External IC | Delays & Holds | Delay a signal one sample period. |
| Unit Delay Resettable | Delays & Holds | Delay a signal one sample period. |
| Unit Delay Resettable External IC | Delays & Holds | Delay a signal one sample period. |
| Unit Delay With Preview Enabled | Delays & Holds | Support calculations that have feedback and depend on the current input. |
| Unit Delay With Preview Enabled Resettable | Delays & Holds | Support calculations that have feedback and depend on the current input. |
| Unit Delay With Preview Enabled Resettable External RV | Delays & Holds | Support calculations that have feedback and depend on the current input. |
| Unit Delay With Preview Resettable | Delays & Holds | Support calculations that have feedback and depend on the current input. |
| Unit Delay With Preview Resettable External RV | Delays & Holds | Support calculations that have feedback and depend on the current input. |
| Wrap To Zero | Nonlinear | Set output to zero if input is above threshold. |

## Renamed Blocks

All of the Fixed-Point Blockset blocks have been renamed for Version 3.1. Old models built with Version 3.0 will continue to work in Version 3.1.

| Version 3.0 Block Name | Version 3.1 Block Name |
|---|---|
| FixPt Absolute Value | Abs |
| FixPt Bitwise Operator | Bitwise Operator |
| FixPt Constant | Constant |
| FixPt Conversion | Conversion |
| FixPt Conversion Inherited | Conversion Inherited |
| FixPt Data Type Propagation | Data Type Propagation |
| FixPt Dead Zone | Dead Zone |
| FixPt Dot Product | Dot Product |
| FixPt Dynamic Look-Up Table | Look-Up Table Dynamic |
| FixPt FIR | FIR |
| FixPt Gain | Gain |
| FixPt Gateway In | Gateway In |
| FixPt Gateway In Inherited | Gateway In Inherited |
| FixPt Gateway Out | Gateway Out |
| FixPt Integer Delay | Integer Delay |
| FixPt Logical Operator | Logical Operator |
| FixPt Look-Up Table | Look-Up Table |
| FixPt Look-Up Table (2D) | Look-Up Table (2D) |
| FixPt Matrix Gain | Matrix Gain |
| FixPt MinMax | MinMax |

| Version 3.0 Block Name | Version 3.1 Block Name |
|---|---|
| FixPt Multiport Switch | Multiport Switch |
| FixPt Product | Product |
| FixPt Relational Operator | Relational Operator |
| FixPt Relay | Relay |
| FixPt Saturation | Saturation |
| FixPt Sign | Sign |
| FixPt Sum | Sum |
| FixPt Switch | Switch |
| FixPt Tapped Delay | Tapped Delay |
| FixPt Unary Minus | Unary Minus |
| FixPt Unit Delay | Unit Delay |
| FixPt Zero-Order Hold | Zero-Order Hold |

## New Functions

The Fixed-Point Blockset 3.1 includes the new functions listed below.

| Function Name | Description |
|---|---|
| `fixpt_look1_func_approx` | Optimize for a fixed-point function, the x values, or breakpoints, that are generated from a lookup table. |
| `fixpt_look1_func_plot` | Plot a function with x values generated by the `fixpt_look1_func_approx` function. |

## Data Type Support

All the Fixed-Point Blockset 3.1 now accept matrix and frame-based signals, except for the following two blocks, which accept only sample-based signals:

- Dot Product
- FIR

If you want to update a model built with Version 3.0 to take advantage of the additional data type support, you should replace all the old blocks in the model with the corresponding Version 3.1 blocks. The section “Renamed Blocks” on page 17-8 lists the Version 3.0 blocks and their corresponding Version 3.1 blocks.

## Changes to Functionality

The following changes have been made to the functionality of the Fixed-Point Blockset 3.1 since Version 3.0:

- The **Doubles Override** parameter in the Fixed-Point Blockset Interface has been renamed to **Datatype Override**, and now includes more options.
- The MultiPort Switch block has a new parameter called **Use zero based indexing**.
- The Switch block has a new parameter called **Criteria for passing first input**, which enables you to control the conditions under which the first input is passed.

# Known Documentation Problem

This section updates the Fixed-Point Blockset 3.1 documentation set, reflecting a known Fixed-Point Blockset 3.1 documentation problem.

## Error in Printed Version of Documentation

The Fixed-Point Blockset 3.1 supports floating-point types, except for custom floating-point types.

In the appendix "Code Generation" of the *Fixed-Point Blockset User's Guide*, the section "Storage Class of Variables," in "Code Generation Support," contains the following incorrect reference to data type support in the blockset:

- No floating-point support except for the fixed-point gateway blocks.

This should be corrected as follows:

- Floating-point types are supported, except for custom floating-point types.

18

# Image Processing Toolbox 3.1 Release Notes


**New Features** . . . . . 18-2
Morphology . . . . . 18-3
Spatial Transformations . . . . . 18-4
Image Registration . . . . . 18-5
Integer Image Arithmetic . . . . . 18-6
Integer Image Filtering . . . . . 18-7
Deconvolution/Deblurring . . . . . 18-7
Support for DICOM Files . . . . . 18-8
Miscellaneous . . . . . 18-8
New Demos . . . . . 18-9

**Major Bug Fixes** . . . . . 18-11

# New Features

This section describes the new features and enhancements of the Image Processing Toolbox, Version 3.0, and the Version 3.1 update, both introduced since the Image Processing Toolbox 2.2.2 (Release 12.0).

---

**Note** The Image Processing Toolbox 3.0 was made available in Web-downloadable form after Release 12.0. The Image Processing Toolbox 3.1 is part of Release 12.1.

---

The new features introduced in the Image Processing Toolbox, Version 3.0, include:

- Many new morphology functions
- New spatial transformation functions
- New image registration functions, with a new graphical user interface
- New integer image arithmetic functions
- New Integer image filtering function
- New image deblurring (deconvolution) functions
- Support for DICOM files
- Miscellaneous new functions
- New image processing demos

The new features introduced in the Image Processing Toolbox, Version 3.1, include:

- New deblurring function, `deconvblind`, that implements the blind deconvolution algorithm
- New utility function, `label2rgb`, that converts a label matrix into an RGB color image

For information about Image Processing Toolbox features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## Morphology

Version 3.0 adds a broad suite of new mathematical morphology tools open up broad new classes of applications in segmentation and image enhancement.

The existing dilation and erosion operators have been extended to work with grayscale images. New functions range from additional basic operators (opening, closing, tophat) to advanced tools useful for segmentation (distance transforms, reconstruction-based operators, and the watershed transform). The functions use advanced techniques for high performance, including automatic-structuring element decomposition, 32-bit binary image packing, and queue-based algorithms.

| Function | Description |
|---|---|
| `bwareaopen` | Binary area open (remove small objects) |
| `bwdist` | Distance transform |
| `bwhitmiss` | Binary hit-miss operation |
| `bwlabeln` | Label-connected components in N-D binary image |
| `bwpack` | Pack binary image |
| `bwulterode` | Ultimate erosion |
| `bwunpack` | Unpack binary image |
| `conndef` | Default connectivity array |
| `imbothat` | Perform bottom-hat filtering |
| `imclearborder` | Suppress light structures connected to image border |
| `imclose` | Close image |
| `imdilate` | Dilate image |
| `imerode` | Erode image |
| `imextendedmax` | Extended-maxima transform |
| `imextendedmin` | Extended-minima transform |

| Function | Description |
|---|---|
| imfill | Fill image regions and holes |
| imhmax | H-maxima transform |
| imhmin | H-minima transform |
| imimposemin | Impose minima |
| imopen | Open image |
| imreconstruct | Morphological reconstruction |
| imregionalmax | Regional maxima |
| imregionalmin | Regional minima |
| imtophat | Tophat filtering |
| strel | Create morphological structuring element |
| strel/getheight | Get structuring element height |
| strel/getnhood | Get structuring element neighborhood |
| strel/getsequence | Get sequence of decomposed structuring elements |
| strel/isflat | Return true for flat structuring element |
| strel/reflect | Reflect structuring element about its center |
| strel/translate | Translate structuring element |
| watershed | Find image watershed regions |

## Spatial Transformations

Version 3.0 adds functions for applying a variety of spatial transformations to images and to points. This is a core computational capability. Supported transform types include affine, projective, and user-defined custom transformations. Multidimensional transformations are supported, where you can control which dimensions are the transform dimensions. For example, you can apply a two-dimensional transform to an RGB image, and each color plane is automatically transformed the same way. You can even control the type of

interpolation independently along each dimension, and specify interpolants that you define.

| Function | Description |
|---|---|
| `checkerboard` | Create checkerboard image |
| `findbounds` | Find output bounds for geometric transformation |
| `fliptform` | Flip the input and output roles of a `TFORM` struct |
| `imtransform` | Apply geometric transformation to image |
| `makeresampler` | Create resampler structure |
| `maketform` | Create geometric transformation structure (`TFORM`) |
| `tformarray` | Geometric transformation of a multidimensional array |
| `tformfwd` | Apply inverse geometric transformation |
| `tforminv` | Apply forward geometric transformation |

## Image Registration

Version 3.0 adds several functions useful for registering (aligning) two images. This is critical in remote sensing and medical imaging, for example. There are functions for inferring various spatial transformations from control-point pairs, for the subpixel adjustment of control-point pair locations, and for normalized cross-correlation. There is also a graphical user interface (GUI) for selecting control-point pairs in a pair of images.

| Function | Description |
|---|---|
| `cp2tform` | Infer spatial transformation from control-point pairs |
| `cpcorr` | Tune control-point locations using cross-correlation |
| `cpselect` | Control-point selection tool (graphical user interface) |

| Function | Description |
|---|---|
| cpstruct2pairs | Convert CPSTRUCT to valid pairs of control points |
| normxcorr2 | Normalized two-dimensional cross-correlation |

## Integer Image Arithmetic

The Image Processing Toolbox 3.1 includes new functions for performing arithmetic on image arrays without converting them to double-precision. In addition to the basic operations (add, subtract, multiply, and divide), there are several key functions (absolute difference, linear combination, and complementation) that cannot readily be implemented in terms of the basic operations.

| Function | Description |
|---|---|
| imabsdiff | Absolute difference of two images |
| imadd | Add two images, or add constant to image |
| imcomplement | Complement image |
| imdivide | Divide two images, or divide image by constant |
| imlincomb | Linear combination of images |
| immultiply | Multiply two images, or multiply image by constant |
| imsubtract | Subtract two images, or subtract constant from image |

## Integer Image Filtering

Version 3.0 added a function for performing filtering on image arrays without converting them to double precision, a significant memory savings in a common operation. You can specify several different boundary padding options. You can also perform higher dimensional filtering.

| Function | Description |
|---|---|
| `imfilter` | Filter 2-D and N-D images |

## Deconvolution/Deblurring

Version 3.0 added support for several fundamental algorithms for the deconvolution (deblurring) of images. All of the functions support multidimensional problems.

| Function | Description |
|---|---|
| `deconvblind` | Deblur image using blind deconvolution algorithm [New with Version 3.1] |
| `deconvlucy` | Deblur image using Lucy-Richardson algorithm |
| `deconvreg` | Regularized deconvolution |
| `deconvwnr` | Wiener deconvolution |
| `edgetaper` | Taper image edges according to PSF |
| `fspecial` | Existing function; added `'disk'` and `'motion'` options |
| `otf2psf` | Convert optical transfer function to point-spread function |
| `psf2otf` | Convert point-spread function to optical transfer function |

## Support for DICOM Files

Version 3.0 adds functions for reading image data and metadata from DICOM files. DICOM is an important file and network interchange standard in the area of medical imaging.

| Function | Description |
|---|---|
| `dicomread` | Read image data from DICOM file |
| `dicominfo` | Read metadata from DICOM file |

## Miscellaneous

Version 3.1 included several new utility functions or previously undocumented utility functions. Most of these were created to support functions in the key feature categories, such as deconvolution.

| Function | Description |
|---|---|
| `circshift` | Shift array circularly |
| `graythresh` | Compute global image threshold using Otsu's method (image enhancement) |
| `im2mis` | Convert image to Java MemoryImageSource |
| `imnoise` | Added support for new noise types: `'poisson'` and `'localvar'` |
| `label2rgb` | Convert label matrix to RGB image [New for Version 3.1] |
| `padarray` | Pad array |
| `regionprops` | Renamed from existing function imfeature; extended to N-D |
| `stretchlim` | Find limits to contrast stretch an image |

## New Demos

The Image Processing Toolbox 3.1 includes the 15 new extended example demos, presented in HTML form.

| Demo Name | Brief Description |
|---|---|
| `ipexconformal` | Explore a Conformal Mapping: illustrates how to use spatial- and image-transformation functions to perform a conformal mapping. |
| `ipexdeconvblind` | Deblurring Images Using the Lucy-Richardson algorithm: illustrates use of the `deconvlucy` function. [New with Version 3.1] |
| `ipexdeconvlucy` | Deblurring Images Using the Lucy-Richardson algorithm: illustrates use of the `deconvlucy` function. |
| `ipexdeconvreg` | Deblurring Images Using a Regularized Filter: illustrates use of the `deconvreg` function. |
| `ipexdeconvwnr` | Deblurring Images Using the Wiener Filter: illustrates use of the `deconvwnr` function. |
| `ipexgranulometry` | Finding the Granulometry of Stars in an Image: illustrates how to use morphology functions to perform granulometry. |
| `ipexmri` | Extracting Slices from a 3-Dimensional MRI Data Set: illustrates how to use the image transformation functions to interpolate and reslice a three-dimensional MRI data set, providing a convenient way to view a volume of data. |
| `ipexnormxcorr2` | Registering an Image Using Normalized Cross-correlation: illustrates how to use translation to align two images. |
| `ipexregaerial` | Registering an Aerial Photo to an Orthophoto: illustrates how to use the Control Point Selection Tool to align two images. |

| | |
|---|---|
| `ipexrotate` | Finding the Rotation and Scale of a Distorted Image: illustrates how to use the `cp2tform` function to get the rotation angle and scale factor of a distorted image. |
| `ipexsegcell` | Detecting a Cell Using Image Segmentation: illustrates how to use dilation and erosion to perform edge detection. |
| `ipexsegmicro` | Detecting Microstructures Using Image Segmentation: illustrates how to use morphological opening and closing to extract large objects from an image. |
| `ipexsegwatershed` | Detecting Touching Objects Using Watershed Segmentation: illustrates use of morphology functions to perform marker-control watershed segmentation. |
| `ipexshear` | Padding and Shearing an Image Simultaneously: illustrates how to use the padding options of the image transformation functions. |
| `ipextform` | Creating a Gallery of Transformed Images: illustrates how to use the `imtransform` function to perform many types of image transformations. |

# Major Bug Fixes

This section describes major bug fixes included in the Image Processing Toolbox, Version 3.0 and Version 3.1.

For information about additional bug fixes that are incorporated from recent releases, see the *Release 12 Release Notes*.

- `fspecial` — Fixed incorrect normalization for the Gaussian filter option.
- `improfile` — Fixed an occasional indexing problem caused by round-off error.
- `rgb2ind` — Fixed a problem that caused `rgb2ind` to produce bad results for very large images.
- Functions that operate on binary input images now treat `NaN`s in a consistent manner. When an input array that is expected to be a binary image contains `NaN` values, the `NaN` value is always treated as 1.

19

# Instrument Control Toolbox 1.1 Release Notes

**New Features** . . . . . 19-2
VXI Block and FIFO Read Operations . . . . . 19-2
Freeing the Serial Port on Windows Platforms . . . . . 19-2
Data Parsing . . . . . 19-2
binblock Data Transfer . . . . . 19-3
Tektronix VISA Support . . . . . 19-3
Enhancements to Existing Functions and Properties . . . . 19-3

**Vendor Driver Requirements and Limitations** . . . . 19-4
Tektronix VISA Limitations . . . . . 19-4

**Upgrading from an Earlier Release** . . . . . 19-5
Events, Callbacks, and Function Handles . . . . . 19-5

# New Features

This section introduces the new features and enhancements in the Instrument Control Toolbox 1.1, added since the Instrument Control Toolbox 1.0 (Release 12.0). For a brief introduction to the Instrument Control Toolbox 1.0, refer to the *Release Notes for Release 12*.

## VXI Block and FIFO Read Operations

For VISA-VXI and VISA-GPIB-VXI objects, you can specify if the VXI register offset increments after data is transferred with the `MemoryIncrement` property.

You can configure `MemoryIncrement` to be `block` or `FIFO`. If `MemoryIncrement` is `block`, the `memread` and `memwrite` functions increment the offset after every read and write operation, and data is transferred from consecutive memory elements. If `MemoryIncrement` is `FIFO`, the `memread` and `memwrite` functions always read from or write to the same memory element. Note that the Instrument Control Toolbox 1.0 supported only block data transfer.

## Freeing the Serial Port on Windows Platforms

The serial port object uses the `javax.comm` package to communicate with the serial port. However, due to a memory leak in `javax.comm`, the serial port object is not released from memory. You can use the `freeserial` function to release MATLAB's hold on the serial port.

`freeserial` is necessary only on Windows platforms. You should use `freeserial` only if you need to connect to the serial port from another application after a serial port object has been connected to that port, and you do not want to exit MATLAB.

## Data Parsing

You can parse formatted data read from your instrument with the `scanstr` function. The data is parsed according to the specified delimiters, and is stored in a cell array as either a double or a string.

## binblock Data Transfer

The binary-block (binblock) format is defined as #<N><D><A> where:

- N specifies the number of digits D that follow.
- D specifies the number of data bytes A that follow.
- A is the data to written to the instrument.

You can transfer binblock data between MATLAB and your instrument with the binblockread and binblockwrite functions.

## Tektronix VISA Support

Tektronix VISA is supported for serial and GPIB interfaces.

## Enhancements to Existing Functions and Properties

### Terminator and EOSCharCode Properties

For serial port and VISA-serial objects, you can configure Terminator to a decimal value ranging from 0 to 127, to the equivalent ASCII character, or to empty ('). For serial port objects, you can also set Terminator to CR/LF or LF/CR.

For GPIB- and VXI-related objects, you can configure EOSCharCode to a decimal value ranging from 0 to 255, or to the equivalent ASCII character.

### Timer Events

Some timer events may not be processed if your system is significantly slowed or if the TimerPeriod value is too small. The minimum TimerPeriod value is now 0.01 second.

### VXI Register Read and Write Functions

The memread, memwrite, mempeek and mempoke functions now handle single precision values.

### CompareBits Property

For VISA-GPIB, VISA-VXI, and VISA-GPIB-VXI objects, the CompareBits property no longer exists since its only supported value is 8.

# Vendor Driver Requirements and Limitations

Driver requirements and limitations associated with the vendors supported by the Instrument Control Toolbox 1.0 are described in the *Release Notes for Release 12*.

## Tektronix VISA Limitations

The Tektronix VISA driver has these known limitations:

- Only serial port and GPIB objects are supported.
- Asynchronous read and write operations for the VISA-GPIB object are not supported. Therefore, the following toolbox functionality is not supported:
  - The `readasync` function
  - The `async` flag for the `fprintf` and `fwrite` functions
  - The `BytesAvailableFcn` and `OutputEmptyFcn` properties

---

**Note** The driver limitations described in this section are restricted to the limitations directly associated with using the Instrument Control Toolbox.

---

# Upgrading from an Earlier Release

This section describes the upgrade issues involved in moving the Instrument Control Toolbox 1.0 (Release 12.0) to the Instrument Control Toolbox 1.1 (Release 12.1).

## Events, Callbacks, and Function Handles

Action properties and action functions are now referred to as callback properties and callback functions. This new terminology is reflected in new names for the associated properties and functions. The general rule for the name changes is to change “Action” to “Fcn” for properties, and “action” to “callback” for functions. For example, `TimerAction` has been renamed `TimerFcn`, and `instraction` has been renamed `instrcallback`.

Additionally, if you want to automatically pass the object and event information to the callback function, then you must specify the function as either a function handle or as a cell array. Note that you can also specify the callback function as a string. In this case, the callback is evaluated in the MATLAB workspace and no requirements are made on the function’s input arguments.

# MATLAB C/C++ Math Library 2.2 Release Notes

**New Features** . . . . . 20-2
New Math Library Functions . . . . . 20-2

**Upgrading from an Earlier Release** . . . . . 20-4
Recompile Your Programs . . . . . 20-4
Changed Features . . . . . 20-4

**Known Software and Documentation Problems** . . . . 20-6
Documentation Error . . . . . 20-6

# New Features

This section introduces the new features and enhancements added in the MATLAB C/C++ Math Library 2.2 since the MATLAB C/C++ Math Library 2.1 (Release 12.0).

For information about MATLAB C/C++ Math Library features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## New Math Library Functions

The following new and modified Math Library functions are available in Version 6.1.

### New Math Library Functions

| Function | Purpose |
|---|---|
| `deval`, `mlfDeval` | Evaluate the solution of a differential equation problem using the output of `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`, `ode23t`, `ode23tb`, or `bvp4c`. |

### Math Library Functions with New or Changed Capabilities

| Function | Enhancement |
|---|---|
| `ode45`, ... `mlfOde45`, ... | New MATLAB syntax `sol = solver(odefun,[t0 tf], y0...)` returns a structure that you can use with the new `deval` and `mlfDeval` functions to evaluate the solution at any point on the interval `[t0,tf]`. |
| `polyeig`, `mlfPolyeig` | New MATLAB syntax `e = polyeig(A0,A1,..,Ap)` returns only the eigenvalues of the specified eigenvalue problem. Use `[X,e] = polyeig(A0,A1,...Ap)` if you also want the eigenvectors. This capability is available in MATLAB 6.0. |
| `qz`, `mlfQz` | New MATLAB syntax `[AA,BB,Q,Z,V,W] = qz(A,B)` returns `W`, the left generalized eigenvectors of `A` and `B`. |

**Math Library Functions with New or Changed Capabilities (Continued)**

| Function | Enhancement |
|---|---|
| `reshape`, `mlfReshape` | New MATLAB syntax `reshape(A,...,[],...)` calculates the length of the dimension specified by the placeholder `[]`. |
| `svd`, `mlfSvd` | You now have the option of specifying just the first two output arguments for `svd`. |

# Upgrading from an Earlier Release

There are no upgrade issues moving from the MATLAB C/C++ Math Library 2.1 (Release 12.0) to the MATLAB C/C++ Math Library 2.2 (Release 12.1).

The following issues are involved in upgrading from the MATLAB C/C++ Math Library 2.0 (Release 11) to the MATLAB C/C++ Math Library 2.2.

## Recompile Your Programs

Existing C/C++ Math Library Version 2.0 hand-written source code is compatible with the Version 2.2 library, but you must recompile your code. Additionally, any M-files which were compiled with Version 2.0 of the MATLAB Compiler must be recompiled with Version 2.2 of the MATLAB Compiler before using them with the Version 2.2 library. If you do not recompile your program, it will produce a run-time error.

## Changed Features

### Empty Arrays

In Version 2.0 of the C++ Math Library, any empty array could be used as an indexed deletion operator. In Version 2.2, you must use the `empty()` function for indexed deletion.

### mbuild Function Now Returns Accurate Error Status

The `mbuild` syntax

```
mbuild myprog.c
```

now throws an error when it encounters an error condition.

The `mex` syntax

```
stat = mbuild('myprog.c')
```

now returns a nonzero value to `stat` when it encounters an error condition.

In the past, on Microsoft Windows platforms, `mbuild` always either successfully exited or returned zero (indicating success), regardless of whether an error had actually occurred.

To ensure code from before Release 12 works properly in Release 12, either use `try/catch` logic to deal with error conditions, or use a form of `mbuild` that returns an error status instead of throwing an error. Specifically

```
try
  mbuild something.c
catch
  disp( something failed );
end
```

or

```
status = mbuild( something.c );
if status ~= 0
  disp( something failed );
end
```

# Known Software and Documentation Problems

This section updates the MATLAB C/C++ Math Library documentation set, reflecting a known documentation problem.

## Documentation Error

In the section “Creating Structures” in the Using the C Math Library documentation, the examples use a routine named `mlfDestroyArray`. The name of this routine should be `mxDestroyArray`.

21

# MATLAB Runtime Server 6.1 Release Notes

**New Features** . . . . . 21-2
Configuration File Generation Tool . . . . . 21-2

**Upgrading from an Earlier Release** . . . . . 21-3
Repeat Compiling, Packaging, and Testing Process . . . . . 21-3

**Known Documentation Problems** . . . . . 21-4

# New Features

This section introduces a new feature added in the MATLAB Runtime Server 6.1 since the MATLAB Runtime Server 6.0 (Release 12.0).

For information about MATLAB Runtime Server features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## Configuration File Generation Tool

The MATLAB Runtime Server now includes a new tool, `makeconfig`, for generating the configuration file for packaging an application.

# Upgrading from an Earlier Release

There are no upgrade issues moving from the MATLAB Runtime Server 6.0 to the MATLAB Runtime Server 6.1.

---

**Note** The upgrade issue described below is involved in moving from the MATLAB Runtime Server 5.0 (Release 11.0) to the MATLAB Runtime Server 6.1.

---

## Repeat Compiling, Packaging, and Testing Process

If you are upgrading a runtime application that you built using the MATLAB Runtime Server 5.0 so that it uses the MATLAB Runtime Server 6.0, then you need to repeat the process of compiling, packaging, and testing the application to ensure that it works properly with MATLAB 6.0. Refer to the *MATLAB Runtime Server Application Developer's Guide* for details.

# Known Documentation Problems

This section updates the MATLAB Runtime Server 6.1 documentation set, reflecting known documentation problems.

These errors occur in the printed, but not the electronic, version of the *MATLAB Runtime Server Application Developer's Guide*:

- The list of files in "Manually Packaging Files for Shipping (PC)" (pages 4-6 and 4-7) is incorrect. Use the electronic version instead.
- If your application uses Java classes, then the discussion about packaging a runtime application (pages 4-3 through 4-7) is incomplete. Use the electronic version of the instructions for automatically or manually packaging your application.
- Pages 3-10 and 3-11 incorrectly refer to the directories `toolbox\runtime\examples\engine\activex` and `toolbox\runtime\examples\engine`, respectively. Both should instead refer to the directory `toolbox\runtime\examples\activex`, which is the actual location of the files for the ActiveX Automation example.
- The electronic version of the "Testing with the Runtime Server Variant" section for runtime engine applications recommends a more robust method, compared to the corresponding section (page 3-8) of the printed version. Also, see the updated procedure for testing the example runtime engine application, and compare with page 3-28 of the printed version.
- The following information about file structures has been added to the electronic version of the "Developing a MATLAB Runtime GUI Application" chapter:

  When creating your deployment structure, you must use the `-inplace` flag with `pcode` or the default `'develStruct'` option of `buildp` to have the P-files mirror the directory structure of the M-files.
- The following information about the packaging utility has been added to the electronic version of the "Shipping a MATLAB Runtime Application" chapter:

  The packaging utility is intended to be used only with the MATLAB portion of your application, however, you can integrate the packaging utility into your overall installation suite.

22

# Neural Network Toolbox 4.0.1 Release Notes

**Major Bug Fixes** . . . . . 22-2
Proper Use of Validation and Test Data . . . . . 22-2
Correct Version 3 Object Loading . . . . . 22-2

# Major Bug Fixes

The Neural Network Toolbox 4.0.1 added several bug fixes, including the following descriptions of particularly important bug fixes.

For information about Neural Network Toolbox 4.0.1 features and bug fixes that are incorporated from recent releases, see the *Release Notes for Release 12*.

## Proper Use of Validation and Test Data

The Neural Network Tool now uses the optional validation and test data properly when training.

## Correct Version 3 Object Loading

Neural network objects saved under Version 3 now load correctly under Version 4.

23

# Optimization Toolbox 2.1.1 Release Notes

**New Features** . . . . . 23-2
Support for Large Problems that Are Not Well-Scaled . . . . 23-2

**Major Bug Fixes** . . . . . 23-3

**Upgrading from an Earlier Release** . . . . . 23-4

# New Features

This section introduces the new features and enhancements added in the Optimization Toolbox 2.1.1 since the Optimization Toolbox 2.1 (Release 12.0).

For information about Optimization Toolbox features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## Support for Large Problems that Are Not Well-Scaled

In Version 2.1.1, large-scale finite differencing is improved numerically to handle cases when an optimization problem is not well-scaled. These changes potentially improve the speed and accuracy of results when using the large-scale versions of `lsqnonlin`, `lsqcurvefit`, `fsolve`, `fmincon` and `fminunc`, particularly if the objective function is not well-scaled.

# Major Bug Fixes

The Optimization Toolbox includes several bug fixes, including the following descriptions of particularly important bug fixes for both Version 2.1 (Release 12.0) and Version 2.1.1 (Release 12.1).

## Upgrading from an Earlier Release

There are no upgrade issues moving from the Optimization Toolbox 2.1 to the Optimization Toolbox 2.1.1.

See the *Release Notes for Release 12* for upgrade issues involved in moving from the Optimization Toolbox 2.0 (Release 11.0) to the Optimization Toolbox 2.1 (Release 12.0).

24

# Power System Blockset 2.2 Release Notes

**New Features** . . . . . 24-2
Powergui Enhancements . . . . . 24-2
New powerlib_extras Documentation . . . . . 24-2

# New Features

This section introduces the new features and enhancements added in the Power System Blockset 2.2 since the Power System Blockset 2.1 (Release 12.0).

For information about Power System Blockset features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## Powergui Enhancements

The Powergui tool in Power System Blockset 2.2 now includes the **Display spectrum of Signal** feature previously available only on certain demos. This feature displays the fast Fourier transform (FFT) of a specified signal.

## New powerlib_extras Documentation

The powerlib_extras block library of Power System Blockset 2.2 consists of miscellaneous, previously undocumented, power blocks. Fifteen of these blocks now have online block reference pages. The other blocks have short block descriptions or block documentation in their masks.

25

# Real-Time Workshop Ada Coder 4.1 Release Notes

**Release Summary** . . . . . 25-2

## Release Summary

This section introduces the new features and enhancements added in the Real-Time Workshop Ada Coder 4.1 since the Real-Time Workshop Ada Coder 4.0 (Release 12.0).

---

**Note** The Real-Time Workshop Ada Coder 4.1 requires the Real-Time Workshop.

---

The Real-Time Workshop Ada Coder 4.1 supports all applicable new features added to the Real-Time Workshop 4.1. These features are described in Chapter 5, “Real-Time Workshop 4.1 Release Notes.”

26

# Real-Time Workshop Embedded Coder 2.0

**Release Summary** . . . . . 26-2

**New Features** . . . . . 26-3
Custom Storage Classes for Data Objects . . . . . 26-3
Enhanced Code Generation Options . . . . . 26-4
Virtualized Output Ports Optimization . . . . . 26-4
Improved HTML Code Generation Report . . . . . 26-4

## Release Summary

Release 2.0 of the Real-Time Workshop Embedded Coder is a major upgrade, incorporating:

- Significant improvements in efficiency and readability of generated code. Many improvements in Real-Time Workshop code generation technology are especially applicable to embedded systems development. These include expression folding and buffer optimizations. For further information on these features, see Chapter 5, "Real-Time Workshop 4.1 Release Notes."
- Custom storage classes for signal, state, and parameter objects, for embedded systems development.
- Additional and enhanced code generation options.
- Improved HTML code generation report.

See "The Real-Time Workshop Embedded Coder" in the *Real-Time Workshop User's Guide* for full documentation of the Real-Time Workshop Embedded Coder.

# New Features

This section introduces the new features and enhancements added in the Real-Time Workshop Embedded Coder 2.0, since the Real-Time Workshop Embedded Coder 1.0.

For information about Real-Time Workshop Embedded Coder features that are incorporated from Version 1.0, see the Release Notes for Release 12.

## Custom Storage Classes for Data Objects

The Real-Time Workshop Embedded Coder 2.0 implements a number of predefined custom storage classes that are useful in embedded systems development. These classes extend the built-in storage classes provided by the Real-Time Workshop. The built-in classes provide limited control over the form of the code generated for references signals, parameters, and other types of data. These storage classes are suitable for a simulation or rapid prototyping environment, but embedded system designers often require greater control over the representation of data.

Using Real-Time Workshop Embedded Coder custom storage classes, you can define and generate constructs such as bit fields or structs from your model, and easily interface data structures to externally written code.

See “Custom Storage Classes” in the *Real-Time Workshop User's Guide* for further information.

---

**Note** To create Simulink data objects with custom storage classes from M-code, you must have a Real-Time Workshop Embedded Coder license. To share data objects of this type with users who do not have the required license, save the objects in a MAT-file which the users can load into the MATLAB workspace.

---

## Enhanced Code Generation Options

### Initialize Floats and Doubles to 0.0

This option lets you control how internal storage for floats and doubles is initialized. You can initialize floats and doubles to the integer bit pattern `0` (all bits off) or set float and double storage explicitly to the value `0.0`.

See "Initialize Floats and Doubles to 0.0" in the *Real-Time Workshop User's Guide* for further information.

### MAT-File Logging Off by Default

In prior releases, the **MAT-file logging** option was on by default. In Real-Time Workshop Embedded Coder 2.0, the **MAT-file logging** option is off by default. We recommend this setting because it eliminates the extra code and memory usage required to maintain logging variables.

See "MAT-file Logging and Code Efficiency" in the *Real-Time Workshop User's Guide* for further information.

## Virtualized Output Ports Optimization

The virtualized output ports optimization lets you eliminate code and data storage associated with root output ports under certain conditions.

See "Virtualized Output Ports Optimization" in the *Real-Time Workshop User's Guide* for further information.

## Improved HTML Code Generation Report

The format of the Real-Time Workshop Embedded Coder code generation report has been enhanced for readability. The "Optimizations" section lists additional options that will better optimize your code. Links from the report to the online Real-Time Workshop Embedded Coder documentation have been expanded.

See "Generating a Code Generation Report" in the *Real-Time Workshop User's Guide* for further information.

27

# Requirements Management Interface 1.0.2 Release Notes

**New Features** . . . . . 27-3
Support for the Latest DOORS Release . . . . . 27-3

# New Features

This section introduces the new features added in the Requirements Management Interface 1.02 since the initial release of the Requirements Management Interface, Version 1.01 (Release 12.0).

For information about the initial release of the Requirements Management Interface, see the *Release Notes for Release 12*.

## Support for the Latest DOORS Release

Version 1.02 works with the latest DOORS release, Version 5.1.

# Signal Processing Toolbox 5.1 Release Notes

**New Features** . . . . . 28-2
Filter Visualization Tool . . . . . 28-2
New Windows . . . . . 28-3
bitrevorder . . . . . 28-4
downsample/upsample . . . . . 28-4
filternorm . . . . . 28-4
goertzel . . . . . 28-4
New Signal Demo . . . . . 28-4

**Enhancements** . . . . . 28-5
FDATool Enhancements . . . . . 28-5
SPTool Enhancement . . . . . 28-5
Enhanced Functions . . . . . 28-6

**Major Bug Fixes** . . . . . 28-7
Bug Fixes Incorporated from Release 12.0 . . . . . 28-7

**Platform Limitations** . . . . . 28-8
FDATool — 640 x 480 Resolution . . . . . 28-8

**Upgrading from an Earlier Release** . . . . . 28-9
Changes to remez . . . . . 28-9

**Known Software Problem** . . . . . 28-10
FDATool and FVTool Pole/Zero Plot . . . . . 28-10
FDATool and FVTool Magnitude and Phase Plot . . . . . 28-10

# New Features

This section introduces the new features added in the Signal Processing Toolbox 5.1 since the Signal Processing Toolbox 5.0 (Release 12.0). Enhancements are described in the next section.

For information about Signal Processing Toolbox features that are incorporated from recent releases, see the *Release Notes for Release 12*.

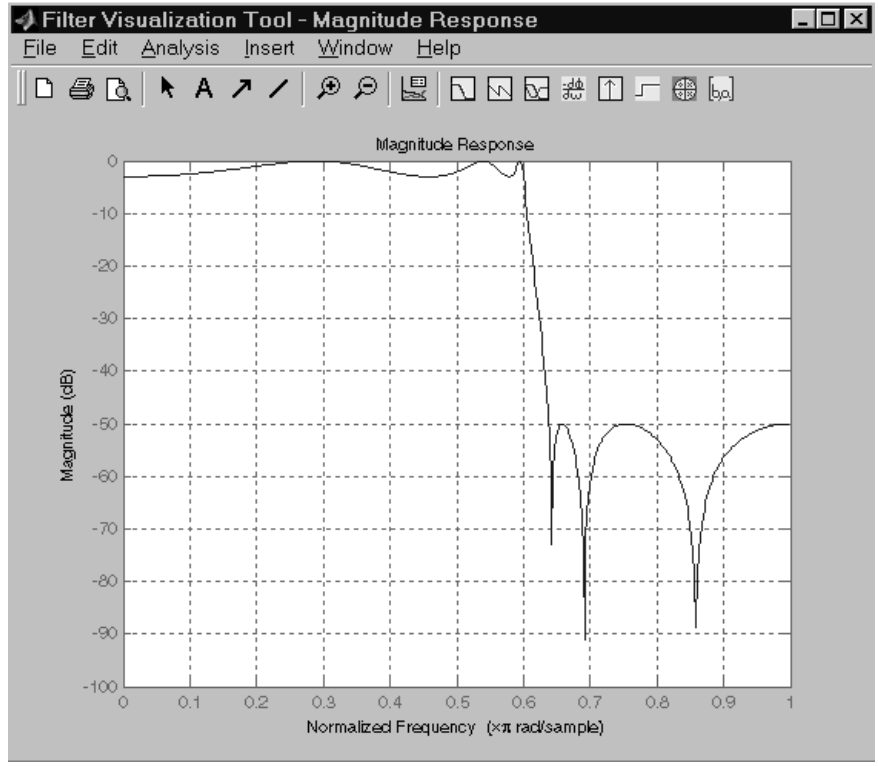
The Signal Processing Toolbox 5.1 provides a number of new features and enhancements, and a completely new facility for analyzing filters, the Filter Visualization Tool.

This section is organized into the following subsections:


- "Filter Visualization Tool" on page 28-2
- "New Windows" on page 28-3
- "bitrevorder" on page 28-4
- "downsample/upsample" on page 28-4
- "filternorm" on page 28-4
- "goertzel" on page 28-4
- "New Signal Demo" on page 28-4


## Filter Visualization Tool

The new Filter Visualization Tool (`fvtool`) groups all the filter analysis functions, such as magnitude, phase, impulse response, etc., into a single window where you can easily switch between analyses, add annotations, and print your filter response.




## New Windows

Six new window functions and a general window function have been added to the Signal Processing Toolbox. One window has been renamed. The new windows are:

- Bartlett-Hann (`barthannwin`)
- Blackman-Harris (`blackmanharris`)
- Bohman (`bohmanwin`)
- Gaussian (`gausswin`)
- Nuttall's Blackman-Harris (`nuttallwin`)
- Tukey (tapered cosine) (`tukeywin`)

The general `window` function is an alternate way to specify any Signal Processing Toolbox window.

The rectangular (`boxcar`) window function has been renamed `rectwin`. For compatibility, `boxcar` will continue to work in this release.

## bitrevorder

The new `bitrevorder` function returns input data in bit-reversed order.

## downsample/upsample

The new `downsample` and `upsample` functions decrease and increase the sampling rate of a signal, respectively.

## filternorm

The new `filternorm` function computes the 2-norm or inf-norm of a digital filter.

## goertzel

The new `goertzel` function implements the Goertzel algorithm, which is an alternate method of computing the DFT.

## New Signal Demo

A new demo, Vector Transition Processing Using Spectral Windows, has been added. This demo is interactive and demonstrates how to smooth the transition between two vectors by using various spectral windows.

# Enhancements

This section describes the Signal Processing Toolbox enhancements and is organized into the following subsections:


- "FDATool Enhancements" on page 28-5
- "SPTool Enhancement" on page 28-5
- "Enhanced Functions" on page 28-6


## FDATool Enhancements

The following items have been added to the FDATool.

- The new Filter Visualization (`fvtool`) can be launched from **Analysis->Full View Analysis**.
- Print Preview is now an icon on the toolbar and it launches the MATLAB standard print preview.
- Clicking on any point on a filter plot line adds a data marker, which displays information about the current point.
- New windows are included in **Method->FIR->Window** and in the Window option in the Window Specifications pane.
- The ability to export filter coefficients to a C-header file (**File->Create Header File**) is included.
- The raised cosine (`firrcos`) filter method is a new option in the Method menu and in the Design Filter pane.
- A plug-in for Code Compose Studio™ IDE to provide the ability to download coefficients from FDATool to the Developer's Kit for TI DSP is included.
- FDATool is integrated with DSP Blockset and is accessible from the Digital Filter Design block.

## SPTool Enhancement

You can print preview and print your filter design in the SPTool Filter Viewer and Filter Designer.

## Enhanced Functions

The following functions have been enhanced in the Signal Processing Toolbox 5.1.

| Function | Enhancement |
|---|---|
| `buffer` | Takes all MATLAB data types in addition to double-precision data |
| `chirp` | Allows you to specify the shape (concave or convex) of a quadratic chirp operation |
| `eqtflength` | Returns the numerator and denominator orders in addition to the numerator and denominator |
| `freqzplot` | Optionally accepts string input (e.g., `'mag'`) in addition to structure input |
| `lpc` | Outputs a vector of prediction error variances in addition to the linear prediction filter coefficients |
| `medfilt1` | Allows you to specify the matrix dimension along which the filter operates |
| `sgolayfilt` | Allows you to specify the matrix dimension along which the filter operates, and it returns row output for row input. This is consistent with `filter` |

# Major Bug Fixes

The Signal Processing Toolbox 5.1 includes several bug fixes, including the following:

- In FDATool Magnitude and Phase plot, zoom now works properly.
- In FDATool, the highpass FIR design method for Least Squares and Equiripple cases now uses the weights correctly when you use the **Specify Order** option.

## Bug Fixes Incorporated from Release 12.0

The Signal Processing Toolbox 5.1 includes several bug fixes that were made in Release 12.0.

# Platform Limitations

## FDATool — 640 x 480 Resolution

On all platforms, FDATool requires a screen resolution greater than 640 x 480.

# Upgrading from an Earlier Release

There are no upgrade issues moving from the Signal Processing Toolbox 5.0 (Release 12.0) to the Signal Processing Toolbox 5.1.

The following upgrade issue is involved in upgrading from the Signal Processing Toolbox 4.3 (Release 11.1) to the Signal Processing Toolbox 5.1.

## Changes to remez

In the Signal Processing Toolbox 5.1, the `remez` function no longer supports the `'m'` option that previously executed an M-file variant of the function.

# Known Software Problem

## FDATool and FVTool Pole/Zero Plot

In the FDATool analysis window, pole-zero plots do not display more than one root at a given point and data markers cannot be displayed for these points.

## FDATool and FVTool Magnitude and Phase Plot

In the Magnitude and Phase plot, you can use data markers only on the phase curve.

29

# Simulink Performance Tools 1.1 Release Notes

**New Features** . . . . . 29-2
Model Coverage Tool . . . . . 29-2
Graphical Merge Tool . . . . . 29-4

**Platform Limitations for HP and IBM** . . . . . 29-6
Limited Support Model Coverage Tool . . . . . 29-6
Graphical Merge Tool Not Supported . . . . . 29-6

# New Features

This section introduces the new features and enhancements added in Simulink Performance Tools 1.1 since Simulink Performance Tools 1.0 (Release 12.0). The following tools were enhanced:


- "Model Coverage Tool" on page 29-2
- "Graphical Merge Tool" on page 29-4


For an introduction to the Simulink Performance Tools, see the *Release Notes for Release 12*.

## Model Coverage Tool

This section describes features and enhancements added to the **Model Coverage Tool**.

### Additional Coverage Types

This version adds the following types of coverage analysis:

- Condition coverage

  Examines blocks that output the logical combination of their inputs, e.g., the Logic block, and Stateflow transitions. A test case achieves full coverage if it causes each input to each instance of a logic block in the model and each condition on a transition to be true at least once during the simulation and false at least once during the simulation.
- Modified condition/decision coverage (MC/DC)

  Examines blocks that output the logical combination of their inputs, e.g., the Logic block, and Stateflow transitions to determine the extent to which the test case tests the independence of logical inputs and transition conditions.
- Look-up table (LUT) coverage

  Examines blocks, such as the Look-Up Table block, that output the result of looking up one or more inputs in a table of inputs and outputs, interpolating between or extrapolating from table entries as necessary. Lookup table coverage records the frequency that table lookups use at each interpolation interval.

See "Coverage Analysis" in the "Model Coverage Tool" section of *Using Simulink* for more information.

### Additional Covered Block Types

This version covers the following additional block types:

- Look-Up Table
- Look-Up Table (2-D)
- Combinatorial Logic
- For
- If
- Logic
- MinMax
- Relay
- SwitchCase
- While

See “Covered Blocks” in the “Model Coverage Tool” section of *Using Simulink* for more information.

### Additional Coverage Reporting Options

This version provides the following additional coverage reporting options:

- Include each test in the model summary
- Produce bar graphs in the model summary
- Use two color bar graphs
- Display hit/count ratio in the model summary
- Don't report fully covered model objects

See “HTML Settings” in the “Model Coverage Tool” section *Using Simulink* for more information.

### Hyperlink Enhancements

When you click a hyperlink to the model in a coverage report, the tool now opens the model, if it is not already open. Hyperlinks to the model in the coverage report now persist across Simulink sessions. This means that you no longer have to regenerate a report in order to use its hyperlinks.

## Graphical Merge Tool

This section describes features and enhancements added to the **Graphical Merge Tool** since Simulink Performance Tools 1.0 (Release 12.0).

### Name Change

The Graphical Merge Tool was known as the Model Differencing Tool in the previous version. The name was changed to reflect the tool's new model merging capability.

### Merging Models

This version not only detects differences between two models but can also optionally merge the differences into either or both models. The merge capability simplifies collaborative development of models. For example, colleagues can each work on separate copies of a model and, when done, use the Graphical Merge Tool to combine the separate versions. See "Merging Model Differences" in *Using Simulink* for more information.

### Additional Model Comparison Options

The Simulink Performance Tools 1.0 allowed you two compare two models that both reside in your file system. The Simulink Performance Tools 1.1 allows you to compare an in-memory version of a model to the version last saved on your file system or to a version stored in a source control system. This allows you to determine what modifications you have made to a model before saving it. You can also compare two versions of a model stored in a source control system to determine how they differ. See "Comparing Models" in *Using Simulink* for more information.

### Show and Hide Parameters Panel

This version allows you to show or hide the tool's **Parameters** panel. Hiding the panel provides more room for the tool's color-coded differences display.

### Single Scrollbar

The previous version provided separate scrollbars to scroll the difference panels for each model being compared. This version provides a single scrollbar that scrolls both panels simultaneously. This keeps the displays in the two panels aligned, making it easier to compare the two models.

### Aligned Model Content Displays

This version aligns the display of the contents of the two models so that an item that appears in both models appears at the same relative position in the content pane for each model. If an item appears in only one of the two models, the corresponding position in the other model's content pane is empty. This visual alignment of the content panels makes it easy to spot differences between the two models. See "Differences Panes" in *Using Simulink* for more information.

### Model History List

This release adds model history lists at the top of each model content pane in the **Graphical Merge Tool**. Each list displays the four models that have most recently appeared in the associated pane. To compare a model on the list to the model that appears in the other content pane, select the model in the history list and press **Return**.

# Platform Limitations for HP and IBM

**Note** The Release 12 platform limitation for the Simulink Performance Tools for the HP and IBM platforms still applies to Release 12.1. That limitation is summarized below.

## Limited Support Model Coverage Tool

On the HP and IBM platforms, you must use the command line commands, for example, `cvsim`, `cvtest`, and `cvhtml`, to generate and display coverage data. The tool displays coverage reports in your system's default browser. The hyperlinks from the report to the model do no work.

## Graphical Merge Tool Not Supported

The **Graphical Merge Tool** is not supported on the HP and IBM platforms.

30

# Simulink Report Generator 1.2 Release Notes

**New Features** . . . . . 30-2
Block Type: Look-Up Table Component Reports on N-D Look-Up Tables . . . . . 30-2
Real-Time Workshop Requirement Removed for Some Components . . . . . 30-2
Image Titles for Snapshot Components . . . . . 30-3

**Upgrading from an Earlier Release** . . . . . 30-4
Use of Existing SGML Files . . . . . 30-4

# New Features

This section introduces the new features and enhancements added in the Simulink Report Generator 1.2 since the Simulink Report Generator 1.1 (Release 12.0).

---

**Note** The Simulink Report Generator extends the functionality of the MATLAB Report Generator.

---

## Block Type: Look-Up Table Component Reports on N-D Look-Up Tables

The Simulink Report Generator 1.2 component Block Type: Look-Up Table allows you to report on the following N-D Simulink look-up tables, in addition to the Look-Up Table and Look-Up Table (2-D) that were supported in earlier releases:

- Look-Up Table (n-D)
- Direct Look-Up Table (n-D)

Note that the table created for the N-D data is a 2-D slice through the N-D data.

## Real-Time Workshop Requirement Removed for Some Components

The **By simulation order** option for sorting blocks no longer requires having the Real-Time Workshop product installed. The components affected by the removal of the Real-Time Workshop requirement for this option are:

- Block Loop
- Chart Loop
- Fixed-Point Block Loop
- Object Summary Table

Also, the Block Execution Order component no longer requires Real-Time Workshop.

## Image Titles for Snapshot Components

You can add titles to images for the System Snapshot and Stateflow Snapshot (chart) components. This is in addition to the caption option that was available in earlier releases.

# Upgrading from an Earlier Release

There are no upgrade issues moving from the Simulink Report Generator 1.1 to the Simulink Report Generator 1.2.

The following upgrade issue is involved in moving from the Simulink Report Generator 1.0 (Release 11.0) to Simulink Report Generator 1.2.

## Use of Existing SGML Files

If you generated a report using the Simulink Report Generator 1.0 and saved the report file as an SGML file, and then try to convert that SGML file using the Simulink Report Generator 1.1 `rptconvert` program, the conversion fails.

In such cases, you need to regenerate the SGML file using the Simulink Report Generator 1.1.

# Stateflow Coder 4.1 Release Notes

**New Features** . . . . . 31-2
Single-Precision Constants in Code Generation . . . . . 31-2

**Major Bugs Fixes** . . . . . 31-3
Code Generation for Default Transitions in Parallel States . 31-3
Code Generation for Double-Precision Whole Numbers . . . 31-3

# New Features

This section introduces the new features and enhancements added in Stateflow Coder 4.1 since the online release of Stateflow Coder 4.0.2 (Release 12).

---

**Note** For information on new features in Stateflow 4.1, see the Stateflow 4.1 Release Notes.

---

For information about the features of Stateflow and Stateflow Coder 4.1 added in prior releases, including Stateflow and Stateflow Coder 4.0 (Release 12), see the *What's New* link from the Stateflow product page on the MathWorks Web site (`www.mathworks.com`).

## Single-Precision Constants in Code Generation

Code generation now emits single-precision constants with a trailing `F` to the C-compiler. This results in lower ROM size. For example, the action language statement `x = y + single(1.375);` now generates the code `x = y + 1.375F;`.

# Major Bugs Fixes

Below is a list of known bugs from Stateflow Coder 4.0.2 that are *fixed* in Stateflow Coder 4.1.

---

**Note** For information on major bug fixes in Stateflow 4.1, see the Stateflow 4.1 Release Notes.

---

## Code Generation for Default Transitions in Parallel States

States with Parallel decomposition with default transition paths generated incorrect code in Stateflow Versions 3.0 through 4.0.2.

## Code Generation for Double-Precision Whole Numbers

Double-precision whole numbers were incorrectly emitted as integers (e.g., 5 instead of 5.0) in code generation in Stateflow Versions 3.0 through 4.0.2.

# System Identification Toolbox 5.0.1 Release Notes

**Upgrading from an Earlier Release** . . . . . 32-2
Syntax of data Function . . . . . 32-2
Theta Models No Longer Supported . . . . . 32-2

# Upgrading from an Earlier Release

There is one new upgrade issue involved in moving from the System Identification Toolbox 5.0 to the System Identification Toolbox 5.0.1.

## Syntax of data Function

The syntax

```
data{...}
```

is no longer supported. Use the functions `getexp` or `merge iddata` instead.

---

**Note** The following upgrade issue is involved in moving from System Identification Toolbox 4.0.5 (Release 11.0) to the System Identification Toolbox 5.0.1.

---

## Theta Models No Longer Supported

Theta models (matrices) are no longer supported in the System Identification Toolbox 5.0. Existing code that uses functions such as `th2par` and `th2ss` to access the theta model data will continue to work in the System Identification Toolbox 5.0. However, if you have code that directly indexes into the theta matrix (e.g., `th(1,3)`), that code will no longer work.

33

# Virtual Reality Toolbox 2.0 Release Notes

**Introduction to the Virtual Reality Toolbox** . . . . . 33-2
Simulink Interface . . . . . 33-2
MATLAB Interface . . . . . 33-2
Virtual Reality Modeling Language (VRML) . . . . . 33-2
VRML Viewing . . . . . 33-3
VRML Authoring . . . . . 33-3

# Introduction to the Virtual Reality Toolbox

The Virtual Reality Toolbox extends the capabilities of MATLAB and Simulink into the world of virtual reality graphics. Using standard Virtual Reality Modeling Language (VRML) technology, you can create animated three-dimensional scenes that are driven from the MATLAB and Simulink environment.

## Simulink Interface

You can observe a simulation of your dynamic system over time in a visually realistic three-dimensional model.

Most of the Virtual Reality Toolbox features can be implemented with Simulink blocks. Once you include these blocks in a Simulink diagram, they allow you to select the virtual world, which you connect to Simulink signals. The Virtual Reality Toolbox automatically scans the virtual world for available VRML nodes that can be driven by Simulink.

## MATLAB Interface

The Virtual Reality Toolbox provides a flexible MATLAB interface to a virtual reality world.

## Virtual Reality Modeling Language (VRML)

The Virtual Reality Modeling Language (VRML) is an ISO standard that is open, text-based, and uses a WWW-oriented format. You use VRML to define a virtual world that you can display in a Web browser and connect to a Simulink model.

The Virtual Reality Toolbox uses many of the advanced features defined in the current VRML97 specification. The term VRML, in this chapter, always refers to VRML as defined in the VRML97 standard ISO/IEC 14772-1:1997. This format includes a description of three-dimensional scenes, sound, internal actions, and WWW anchors.

## VRML Viewing

If you install a VRML plug-in, you can view a virtual world in your preferred Web browser. The Virtual Reality Toolbox includes the popular VRML plug-in, blaxxun Contact.

The Virtual Reality Toolbox connects MATLAB and Simulink with a VRML enabled browser to display a simulated process using the TCP/IP protocol. This allows you to watch a simulated virtual world not only on the computer where MATLAB and Simulink are running, but also on other computers connected through the Internet.

## VRML Authoring

The Virtual Reality Toolbox includes the leading VRML authoring tool, V-Realm Builder, by Ligos Corp. With the addition of this VRML authoring tool, the Virtual Reality Toolbox provides a complete authoring, development, and working environment for carrying out three-dimensional visual simulations.

## Updated Documentation Coming in Printed and Web-Downloadable Form

The Virtual Reality Toolbox 2.0 documentation included on the Release 12.1 CD represents an early version of the product documentation. This documentation is being updated and will be posted to the MathWorks Web site, where you can download it. When the updated documentation is ready, a printed version will be shipped to all Virtual Reality Toolbox customers.

34

# Wavelet Toolbox 2.1 Release Notes

**New Features** . . . . . 34-2
disp . . . . . 34-2
wavefun2 . . . . . 34-2
GUI Improvements . . . . . 34-2

**Major Bug Fixes** . . . . . 34-3

**Upgrading from an Earlier Release** . . . . . 34-4

# New Features

This section introduces the new features and enhancements added in the Wavelet Toolbox 2.1 since the Wavelet Toolbox 2.0 (Release 12.0).

For information about Wavelet Toolbox features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## disp

The new `disp` method displays information for a WPTREE object.

## wavefun2

The new `wavefun2` function returns the 2-D scaling and wavelet functions from tensor products of 1-D scaling and wavelet functions.

## GUI Improvements

The GUI menus have been reorganized and improved. See Appendix A, "GUI Reference" in the *Wavelet Toolbox User's Guide* for more information.

# Major Bug Fixes

The Wavelet Toolbox includes several bug fixes, including the following descriptions (online only) of particularly important bug fixes.

## Upgrading from an Earlier Release

There are no upgrade issues moving from Wavelet Toolbox 2.0 (Release 12.0) to the Wavelet Toolbox 2.1.

See the *Release Notes for Release 12* for upgrade issues involved in moving from the Wavelet Toolbox 1.2 (Release 11.0) to the Wavelet Toolbox 2.1.

35

# xPC Target 1.2 Release Notes


**New Features** . . . . . 35-2
C API . . . . . 35-2
UDP Driver Support . . . . . 35-2
Asynchronous Event (Interrupt) Support . . . . . 35-2
CAN FIFO Support . . . . . 35-3
100 Mbit/Second Ethernet Support . . . . . 35-3
Fixed-Point Kernel Support . . . . . 35-3
Increased Data Upload Speed . . . . . 35-3
I/O Device Driver Support . . . . . 35-3

**Known Software and Documentation Problems** . . . . 35-4
Uploading Data with External Mode . . . . . 35-4
Target Command Line Interface . . . . . 35-4
Memory-Mapped Devices . . . . . 35-4
Selecting a Working Directory Inside the MATLAB Root Directory . . . . . 35-4

# New Features

This section introduces the new features and enhancements added to xPC Target 1.2 and xPC Target Embedded Option 1.2 since Version 1.1 of those products (Release 12.0).

For information about xPC Target and xPC Target Embedded Option features that are incorporated from recent releases, see the *Release Notes for Release 12*.

## C API

A beta version of the xPC Target C API is available from the Product News page at `http://www.mathworks.com/support/author/xpc/index.shtml`.

Use the C API to create stand-alone graphical user interfaces (GUIs) or console applications to interact with a real-time target application. xPC Target includes examples for Visual C/C++ and Visual BASIC.

## UDP Driver Support

Communicate with another computer using User Datagram Protocol (UDP) packets. Add UDP driver blocks to your Simulink model and configure them for sending and receiving messages from another system.

The UDP driver blocks are not included on the R12.1 CD. Please check the Product News page at `http://www.mathworks.com/support/author/xpc/index.shtml` for the drivers when they become available.

## Asynchronous Event (Interrupt) Support

Create subsystems that execute asynchronously when an event (interrupt) occurs.

These subsystems can communicate and be part of a larger model that executes with a monotonic step size.

We expect to further enhance the Asynchronous Event drivers currently available on the R12.1 CD. Please check the Product News page at `http://www.mathworks.com/support/author/xpc/index.shtml` for the enhanced version.

## CAN FIFO Support

Use CAN FIFO model I/O drivers to either monitor an entire CAN network, or to minimize driver latency for certain applications.

## 100 Mbit/Second Ethernet Support

Use 10 or 100 megabit/second communication between the host and target computers for downloading real-time application and uploading signal data. The communication uses the TCP/IP protocol with an Ethernet card in both the host and target computers.

## Fixed-Point Kernel Support

The xPC Target kernel runs on PCs (CPUs) that do not have a hardware floating-point unit. If needed, xPC Target uses a software floating-point emulator.

## Increased Data Upload Speed

Upload log data about five times faster or better. Upload scope data about three times faster.

## I/O Device Driver Support

xPC Target 1.2 includes new device drivers blocks:

- New blocks to support additional I/O boards
- Blocks to support a GPIB fieldbus using a National Instruments GPIB controller and an RS232 connection to the target PC
- Blocks to support CAN and CAN FIFO networks

# Known Software and Documentation Problems

This section updates the xPC Target 1.2 and xPC Target Embedded Option 1.2 documentation set, reflecting known xPC Target and xPC Target Embedded Option software and documentation problems.

## Uploading Data with External Mode

xPC Target does not support uploading data to normal Simulink scope blocks when using Simulink external mode. Instead, use xPC Target scope blocks or the xPC Target graphical user interface (GUI).

## Target Command Line Interface

Visualizing or changing parameter values is limited to scalar parameters.

## Memory-Mapped Devices

Some supported boards in the xPC Target I/O library are memory-mapped devices. These memory-mapped boards are accessed in the address space between 640K and 1M in the lower memory area. xPC Target reserves a 112 kB memory space for memory mapped devices in the address range

```
C0000 - DBFFF
```

Some drivers for memory-mapped devices allow you to select an address range supported by the device, but not supported by xPC Target. For example, the CAN drivers for Softing allow you to select memory ranges above DBFFF. Select a memory range supported by both the device and xPC Target.

## Selecting a Working Directory Inside the MATLAB Root Directory

Before you run the command `xpctest`, select a current directory outside of the MABLAB root directory.

During the build process, Real-Time Workshop does not allow files to be saved within the MATLAB root. If you select a current directory within the MATLAB root, the xPC Target test procedure will fail when trying to build a target application.